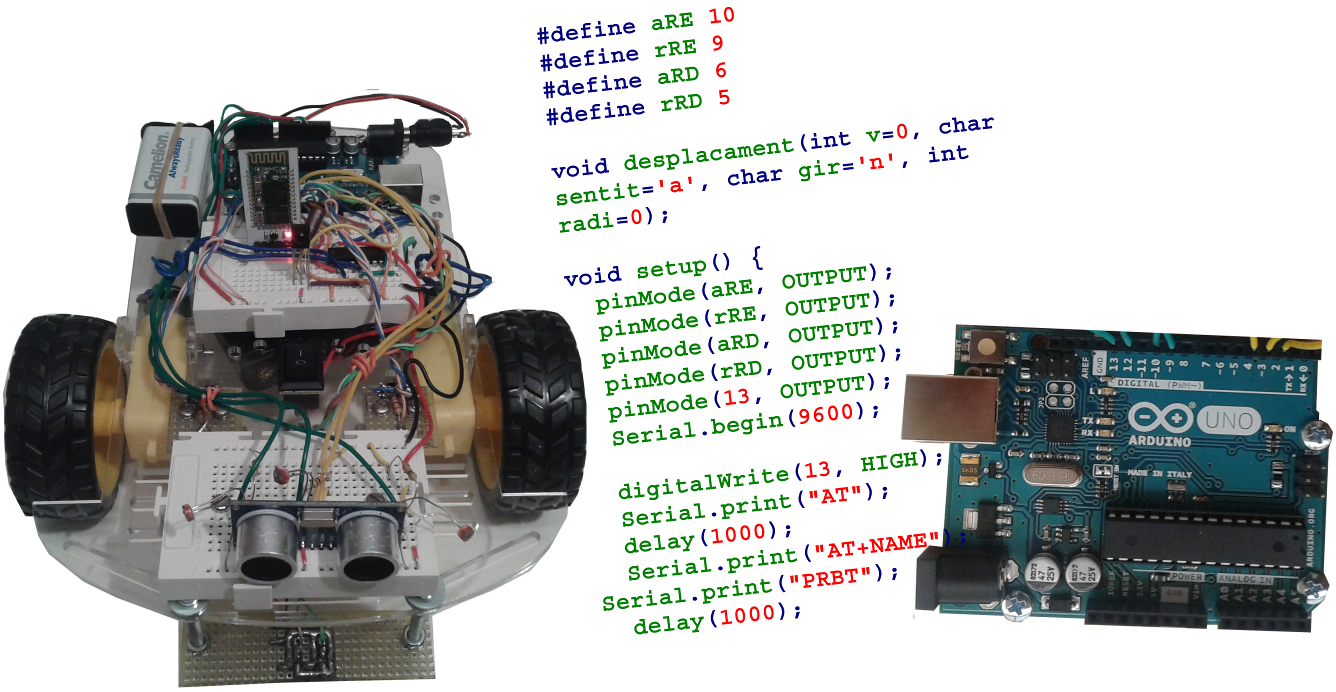
Introducció a l'electrònica, la programació i pràctiques amb

Arduino



(c) Artur Guillamet Sabaté

v1.0 (Novembre, 2015)

Introducció a l'electrònica, la programació i pràctiques amb Arduino

INDEX

	Pàg.
1.- Introducció.	3
2.- Arduino	3
3.- Electrònica. Components electrònics.	6
3.1.- Protoboard	6
3.2.- Resistors.	7
3.3.- Polsadors i interruptors.	10
3.4.- Condensadors.	12
3.5.- Diodes. Diodes Led.	14
3.6.- Transistor bipolar i Mosfet.	17
3.7.- Relés.	21
3.8.- Circuits convertidors A/D i D/A.	24
3.9.- Modulació per amplada de pols, PWM.	25
3.10.- Motor cc, motor pas a pas i servomotor.	26
3.11.- Optoacobladors.	30
4.- Programació en C.	31
4.1.- Estructura bàsica dels programes per Arduino.	32
4.2.- Llibreries.	33
4.3.- Tipus de dades. Constants i variables. Operadors.	33
4.4.- Estructures bàsiques de programació.	34
4.5.- Funcions.	36
4.6.- Funcions predefinides.	37
4.7.- Mètodes Polling i Interrupt.	38
5.- Pràctiques.	39
5.1.- Pràctica 1. Semàfor.	39
5.2.- Pràctica 2. Semàfor controlat d'un pas de vianants.	41
5.3.- Pràctica 3. Termòstat electrònic.	43
5.4.- Pràctica 4. Sensor de llum.	45
5.5.- Pràctica 5. Llum multicolor. So multi to.	48
5.6.- Pràctica 6. Rellotge digital i alarma amb visualitzador LCD.	51
5.7.- Pràctica 7. Dau electrònic.	54
5.8.- Pràctica 8. Control de potència.	57
5.9.- Pràctica 9. Control d'un servomotor.	59
5.10.- Pràctica 10. Control motor c.c.	61

INDEX (cont.)

	Pàg.
5.11.- Pràctica 11. Control d'un motor pas a pas.	63
5.12.- Pràctica 12. Mòbil. Plataforma i control motors.	66
5.13.- Pràctica 13. Mòbil. Seguidor de llum.	71
5.14.- Pràctica 14. Mòbil. Seguidor de línies.	73
5.15.- Pràctica 15. Mòbil. Control amb comandament a distància de TV.	75
5.16.- Pràctica 16. Mòbil. Detector d'obstacles.	78
5.17.- Pràctica 17. Mòbil. Control rotacions motor amb barrera de llum.	80
5.18.- Pràctica 18. Mòbil. Comunicar Arduino per Bluetooth. Control des de ordinador PC i Smartphone Android.	85
5.19.- Pràctica 19. Sensor de so.	92
6.- Exercicis.	94

1.- Introducció.

En aquest dossier és pretén fer una introducció bàsica a la implementació de circuits electrònics programables mitjançant la utilització de circuits microcontroladors.

Per això es fa una introducció a l'electrònica i s'estudia de forma bàsica els principals components electrònics i la programació en el llenguatge C necessaris per poder assolir els coneixements bàsics per realitzar les pràctiques resoltes i exercicis proposats al llarg del desenvolupament d'aquest dossier.

No és pretén donar uns coneixements profunds i exhaustius de les àrees de coneixement sobre l'electrònica, electrònica digital i la programació amb llenguatges d'alt nivell, però, si els coneixements suficients per interpretar adequadament les pràctiques resoltes i els exercicis proposats.

2.- Arduino.

Arduino és un circuit electrònic d'utilització universal. És a dir, esta dissenyat per ser la base de altres dissenys en els que afegirem els circuits electrònics addicionals necessaris i la programació adequada, per aconseguir l'objectiu desitjat.

Aquest circuit ha estat desenvolupat des de l'any 2006 com a open hardware, es a dir, el seu disseny és públic i el pot implementar qualsevol que vulgui. Inicialment es va desenvolupar per part d'uns professors (Massimo Banzi, entre altres) d'un institut de disseny interactiu d'Ivrea, al nord d'Itàlia amb l'objectiu d'obtenir un producte de baix cost útil per l'aprenentatge dels seus alumnes. El seu nom prové del nom d'un bar on aquests professors assistien en les seves hores lliures. Posteriorment, gràcies a la seva naturalesa open hardware, el seu desenvolupament, millora i versions, ha estat promogut per múltiples professors i investigadors en aquesta àrea de coneixements d'arreu del món, sempre mantenint el seu caràcter educatiu i econòmic.

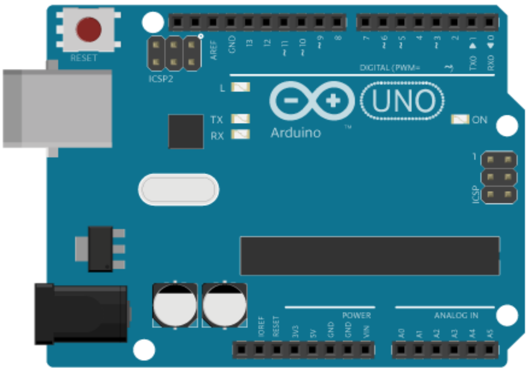
Existeixen diversos models de circuits Arduino, com ara el Arduino Duemilanove, Arduino Ethernet, Arduino Uno, Arduino Mega, Arduino Mini, Arduino Nano, etc. Les principals diferències entre ells consisteix en el tipus de microcontrolador utilitzat, les seves dimensions i la incorporació, o no, d'alguns dispositius perifèrics addicionals.

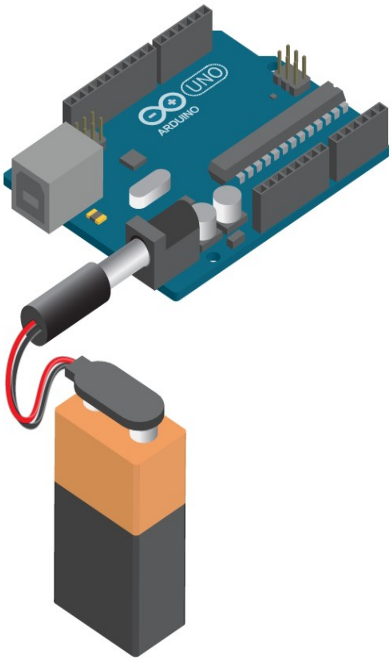
La placa Arduino incorpora tota la circuiteria addicional necessària per facilitar la interconnexió amb altres components electrònics externs i, també, per que funcioni el microcontrolador com ara, els circuits electrònics de Reset, de Clock, d'obtenció i regulació de les tensions adequades (5v i 3.3v) a partir de l'alimentació elèctrica aplicada (de 7v a 12v en cc), circuits de intercomunicació mitjançant el port USB, o el port Ethernet, etc. segons s'escaigui.

En aquest dossier es descriuran les pràctiques utilitzant el model Arduino Uno R3, encara que pel nivell dels exercicis pràctics implementats també es compatible amb altres models de circuits Arduino.

El microcontrolador és un xip electrònic basat en les característiques d'un microprocessador amb l'afegit, en el mateix xip, de unitats de memòria, de clock, unitats de entrada/sortida de diferents tipus, timers, comptadors, gestors de interrupcions, etc. Això confereix al microcontrolador unes característiques pròpies d'un sistema informàtic en un sol xip.

El microcontrolador que incorpora el Arduino Uno R3 és el Atmega328P del fabricant ATMEL AVR.

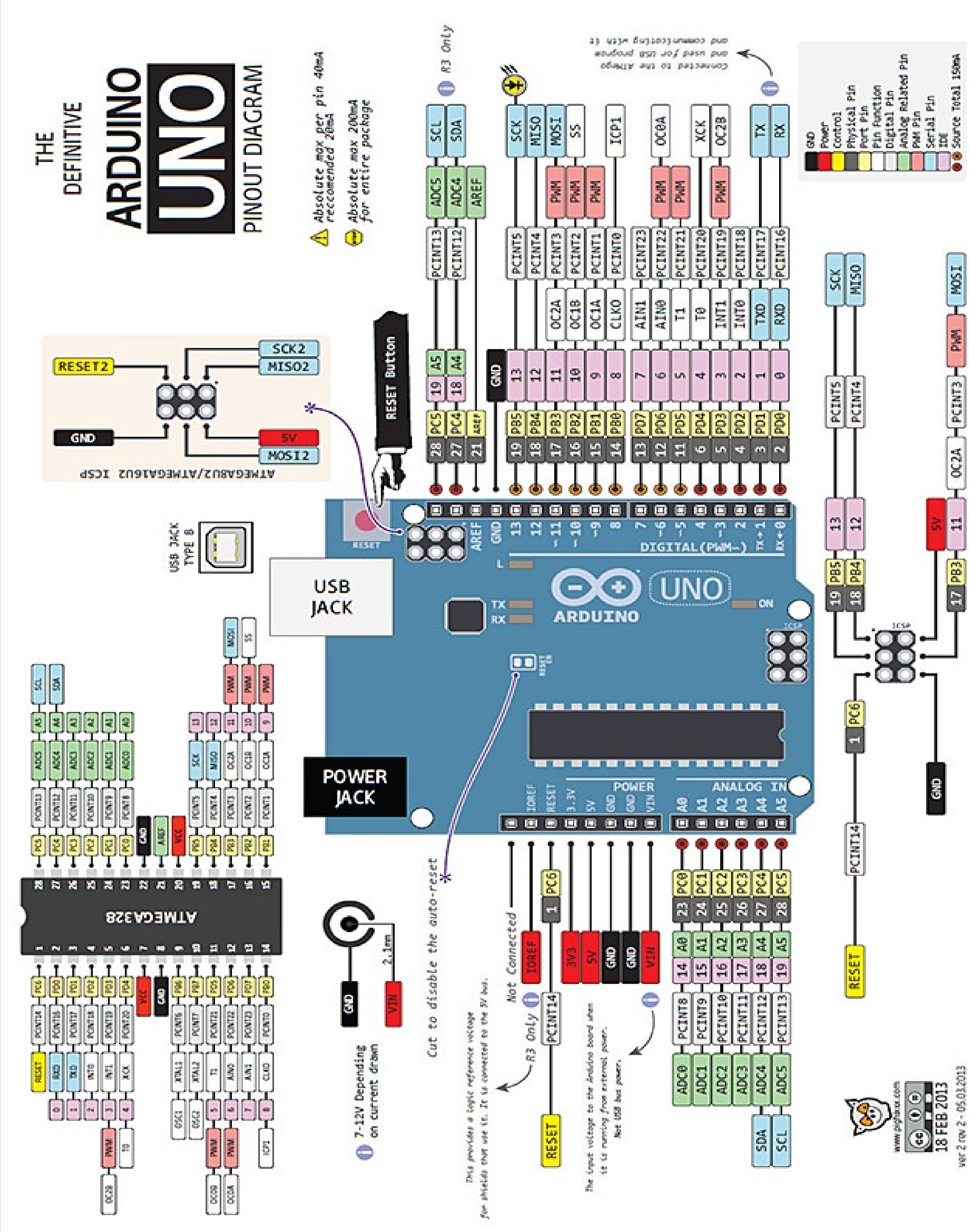




El circuit Arduino s'alimenta amb una tensió d'entre 7v i 12v de cc. Aquesta és pot subministrar de diverses formes:

- Mitjançant el mateix connector USB, si està connectat a un ordinador.
- Mitjançant una pila o una font d'alimentació connectats al connector d'alimentació tal com es veu a la imatge anterior.
- Mitjançant una tensió aplicada entre algun dels pins GND (-) i el pin Vin (+).

Diagrama de pins del microcontrolador ATMEGA 328 i del circuit Arduino Uno que l'incorpora.



Arduino incorpora en el mateix microcontrolador un petit "programa carregador" que li permet carregar els programes creats per l'usuari des del mateix ordinador on s'hagi connectat per USB. Per això es disposa d'un petit programa IDE (Integrated Development Environment, Entorn integrat de desenvolupament), compatible amb Linux, Mac i Windows, que permet escriure programes en un llenguatge d'alt nivell gairebé igual al llenguatge de programació C.

Els programes en llenguatge C escrits en l'entorn IDE de Arduino són compilats a les instruccions de codi màquina específiques del microcontrolador Atmega328P i trameses pel cable USB a aquest microcontrolador per a la seva execució.

Gràcies a la seva concepció de open hardware, s'ha creat múltiples eines que faciliten el treball i desenvolupament amb Arduino, per part de altres persones que col·laboren en el projecte, per tant, a la xarxa podem trobar molta informació i eines adaptables que milloren les possibilitats i versatilitat del producte.

La web oficial dels productes Arduino és: https://www.arduino.cc

3.- Electrònica. Components electrònics.

L'electrònica és una àrea de la tecnologia que s'encarrega del tractament, control i transmissió de la informació mitjançant els fenòmens elèctrics i òptics. Se li diu la tècnica de control dels corrents febles (en contraposició amb les tècniques de control de les corrents fortes, implementades en l'electrotècnia i orientades a la gestió de l'energia elèctrica).

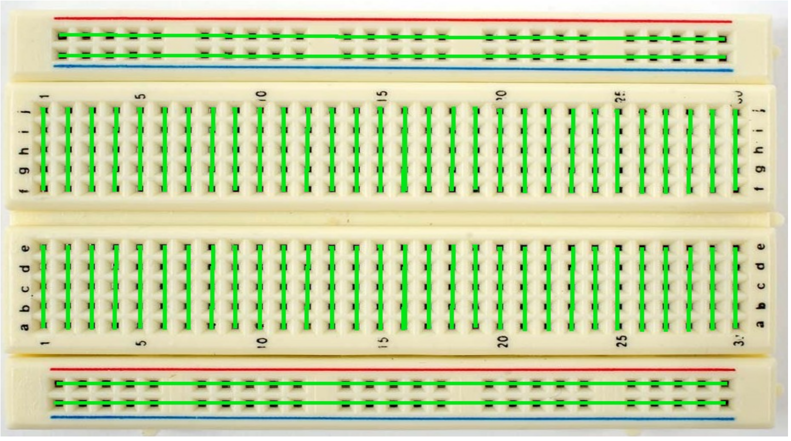
L'electrònica es classifica en dos branques: l'electrònica analògica i la digital. Tot i que totes dues és poden implementar en circuits per si soles, és molt habitual l'existència de circuits híbrids (analògics i digitals) ja que, avui en dia, el tractament de la informació és realitza de forma digital però, les sensacions i senyals d'on s'obté, i transmet, la informació del nostre entorn i els nostres sentits, són tots ells de naturalesa analògica.

En els apartats següents s'analitzarà alguns components electrònics i algunes funcions electròniques que serà necessari conèixer en les implementacions pràctiques d'aquest dossier.

3.1.- Protoboard.

La placa de proves "Protoboard" serveix per provar prototips de circuits electrònics sense que sigui necessari la creació del circuit imprès de suport i connexió. És vàlida per a petits circuits de prova, previ a la construcció del circuit imprès.

Els borns de connexió estan interconnectats internament de la forma que es veu les línies de color verd en la següent imatge:



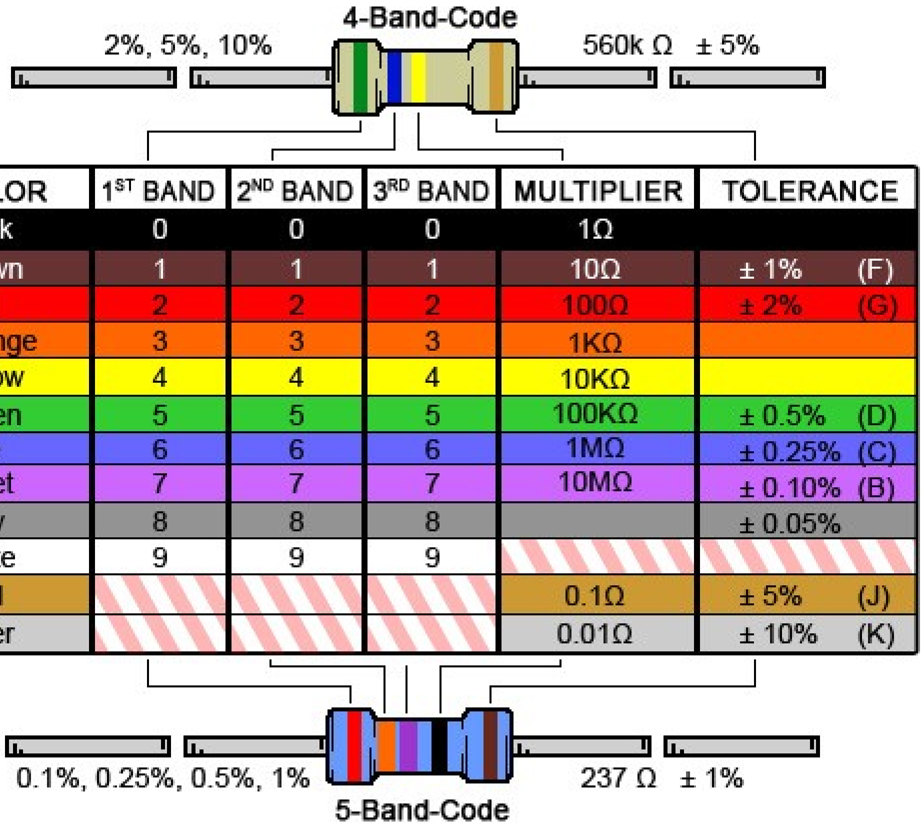
3.2.- Resistors.

Els resistors són els components que introdueixen una oposició resistiva en els circuits electrònics. Amb ells es compleix la coneguda Llei d'Ohm i ens marquen la relació que existeix en un circuits entre la tensió (V) i la intensitat (I) corresponent al paràmetre físic Resistència (R) ($R=\frac{V}{I};V=R\cdot I;I=\frac{V}{R}$). També s'anomenen Resistència de forma col·loquial, fent referència al paràmetre físic que implementen. El seu símbol habitual és: o , es representa amb la lletra R i és mesura amb la unitat: Ohm (Ω, KΩ i MΩ).

Segons la seva fabricació, els resistors fixes poden classificar-se en:

- Resistors bobinats de fil resistiu de potència.
- Resistors de pel·lícula conductora de carboni, metall o òxids metàl·lics.
- Resistors conglomerats de carboni.

Els resistors fixes utilitzats en circuits electrònics poden indicar el seu valor òhmic numèricament o mitjançant un codi de franges de colors pintat a sobre d'ell. El codi és el següent:



COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

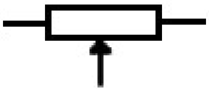
En el codi podem veure els dos casos habituals, resistors amb 4 franges i resistors amb 5 franges. S'observarà que totes les franges estan separades amb la mateixa distància entre elles, excepte una que es troba a una distància més gran de les altres. Aquesta franja amb més separació és la Tolerància. Llavors, descomptant la franja de la Tolerància que ubiquem a la nostra dreta, comencem a comptar les franges indicatives del valor des de l'esquerra i trobarem inicialment 2 o 3 franges indicatives del valor per a la 1ª xifra significativa, 2ª xifra significativa i 3ª xifra significativa (aquesta darrera solament en el cas de resistors de 5 franges) i seguidament una altra franja indicadora del nombre de zeros a ubicar després de les xifres significatives.

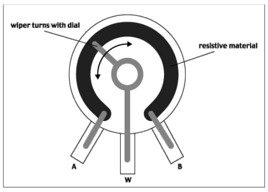
Resistors variables:
Els resistors vists fins ara són de valor fix, és a dir, el valor òhmic no canvia mai. Però, també existeixen els resistors variables, als quals es pot variar la seva resistència òhmica mitjançant un accionament mecànic.

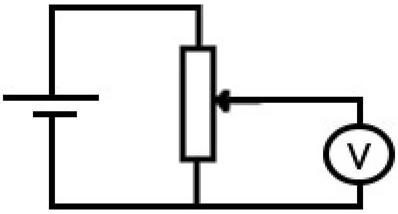
Resistors variables {
- Resistor ajustable
- Potenciòmetre

- **Resistor ajustable.-** És un resistor on es pot modificar el seu valor, però, habitualment no ho poden fer els usuaris convencionals, sinó, els tècnics mitjançant un tornavís petit per a fer-ne l'ajust. Quan la potència del resistor ajustable és elevada també se l'anomena reòstat.

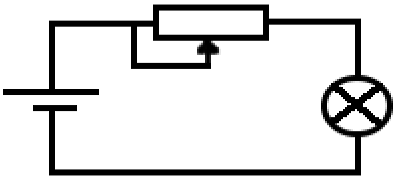
- **Potenciòmetre.-** És una resistència variable que consta de tres borns, els laterals corresponen als extrems de la resistència i el central correspon al cursor mòbil que pot connectar sobre el resistor en una posició seleccionable entre els dos extrems.

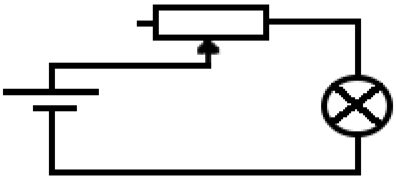






Connexionat com a potenciòmetre (divisor de tensió).





Connexionat com resistor variable (reòstat).

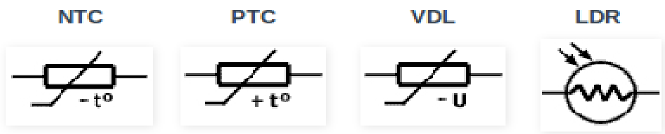
Els resistors variables poden connectar-se com a divisor de tensió (potenciòmetre) o com a reòstat.

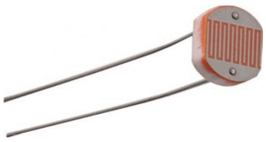
Resistors dependents:
Aquest tipus de resistors tenen un valor òhmic que depèn de un paràmetre físic que l'afecta. Algun exemple d'aquest tipus de resistor el tenim classificat en el quadre següent:

Resistors dependents d'un paràmetre físic {
- Temperatura (NTC i PTC, termistor)
- Llum (LDR, fotoresistència, fotocèl·lula)
- Voltatge (VDR, varistor)

- **NTC.-** coeficient de temperatura negatiu, augmenta la R en disminuir la Tª.
- **PTC.-** coeficient de temperatura positiu, augmenta la R en augmentar la Tª.
- **LDR.-** Resistor depenent de la llum. Anomenat, també, fotocèl·lula o fotoresistor. Disminueix el valor de la R quant incideix llum sobre la LDR. És a dir, amb poca llum tenen una resistència molt alta i quan són il·luminats, el seu valor disminueix molt.

- **VDR.-** Resistor depenent del voltatge. Disminueix el valor de la seva R quant augmenta el valor de la tensió en els seus borns.




LDR, fotoresistor

NTC, termistor 0'5w, 100kΩ

PTC, termistor 0'8Ω

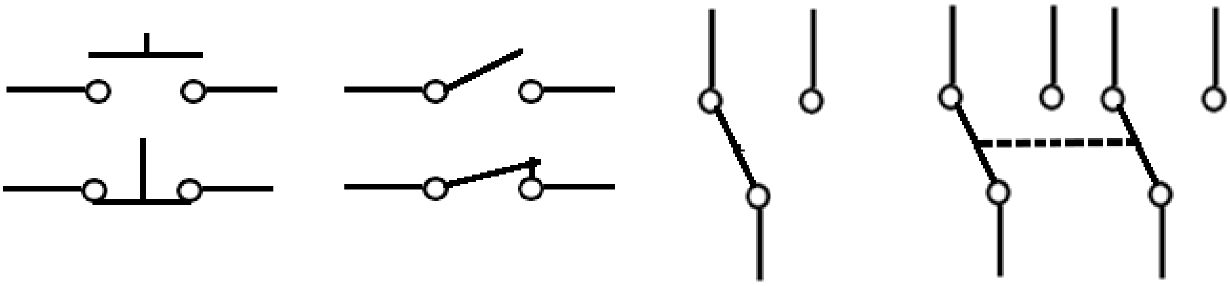
VDR, varistor 250v

3.3.- Polsadors i interruptors.

Els polsadors i interruptors són els dispositius mecànics que serviran per obrir i tancar circuits. El seu accionament pot ser manual o mitjançant un altre dispositiu o sistema que els accioni, com ara, una bobina d'un relé, un final de cursa, un camp magnètic, etc.

Els polsadors tenen una posició estable i els interruptors tenen dos posicions estables.

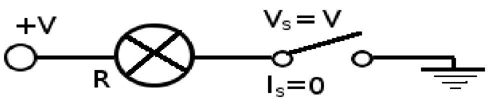
Els més habituals són el polsador, l'interruptor i el commutador, els dos primers poden ser NO (normalment obert) o NT (normalment tancat) i els podem trobar individualment o bé, formant un conjunt units mecànicament de forma solidària, on l'accionament és simultani per a tots ells.

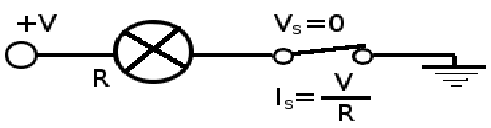


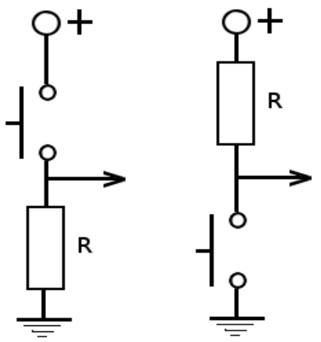
Polsadors NO i NT, Interruptors NO i NT, commutador i commutadors units solidàriament

També existeixen altres tipus de interruptors especials com els interruptors d'inclinació (tilt switch), de moviment o vibració.

Quan l'interruptor del circuit esta en estat obert, tenim que: V>0 i I=0, i en estat tancat: V=0 i I>0.





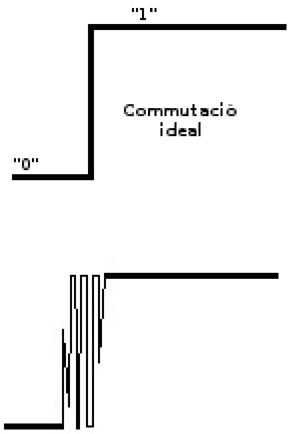


Modes Pull-Down i Pull-Up, respectivament.

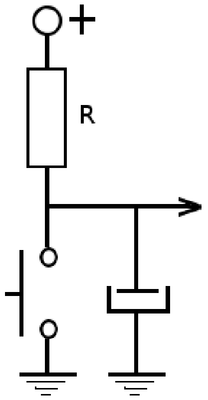
En el circuit previ, al obrir l'interruptor, un dels borns de R queda sense connectar-se, "a l'aire", això no es cap problema en aquest cas, però, en moltes ocasions, en els circuits electrònics no podem deixar els borns "a l'aire", sinó que els hi hem de garantir sempre un potencial elèctric definit, llavors, per aconseguir això, s'implementa l'interruptor conjuntament amb una resistència, que es connectarà a positiu o negatiu segons es vulgui tenir una lògica positiva o negativa i segons ho permeti el circuit.

La commutació en un contacte elèctric és el canvi d'un estat a un altre.
L'accionament mecànic dels interruptors o polsadors per la commutació, provoca l'existència de microconnexions entre els borns que originen el que es coneix com a "rebots" de impulsos elèctrics que tenen una durada de l'ordre de µs o alguns ms. En circuits elèctrics convencionals, això no té cap afectació, però, segons sigui la seva aplicació en un circuit electrònic, pot afectar el correcte funcionament del circuit.

Per absorbir aquestes microconnexions i evitar l'afectació al circuit electrònic sensible a aquest efecte, es fa mitjançant la connexió d'un condensador en paral·lel amb l'interruptor o polsador.

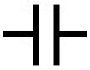
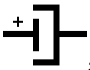
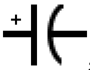
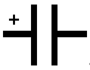


Microconnexions (rebots) que s'estableixen en la commutació d'un contacte mecànic previ a l'establiment de la connexió efectiva.



En cas que els rebots afectin al circuit, podem aplicar un condensador per evitar-ho.

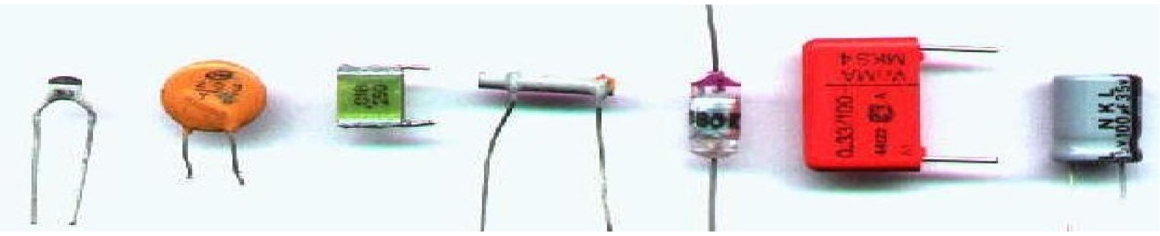
3.4.- Condensadors

Els condensadors són uns components electrònics que acumulen energia elèctrica en forma de tensió. La seva capacitat d'acumular l'energia elèctrica es mesura en Faradis (F) i són habituals els submúltiples pF (10^{-12}), nF (10^{-9}), µF (10^{-6}) i mF (10^{-3}). El símbol convencional del condensador és:  però, en el cas de condensadors polaritzats també tenim els símbols: , , .

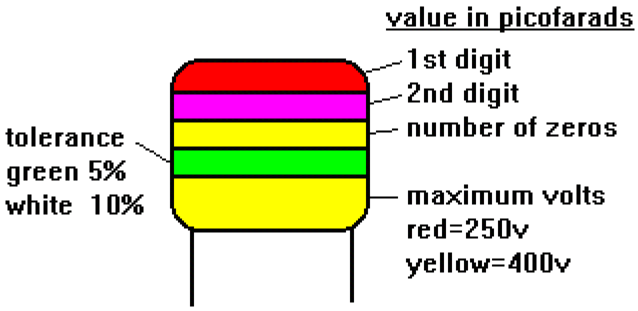
Els condensadors es classifiquen segons el tipus de material dielèctric que es troba entre les dues plaques enfrontades. Aquest material dielèctric pot ser aire o un material sòlid. Així poden tenir una classificació dels condensadors segons el material dielèctric emprat: d'aire, paper, ceràmica, vidre, polièster, alumini, tàntal, etc.

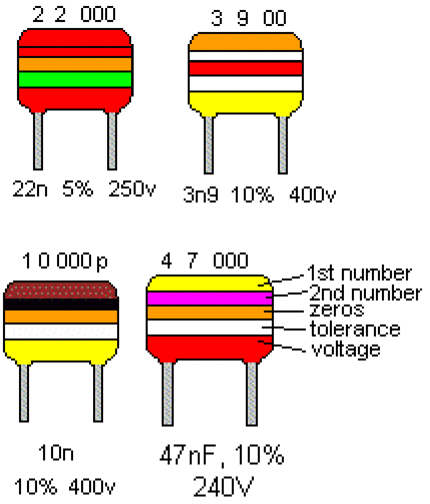
També es classifiquen en polaritzats i no polaritzats. De la llista anterior, els de alumini i els de tàntal són polaritzats, la resta són sense polarització. Els condensadors no polaritzats poden funcionar tan en c.c. com en c.a., en canvi, els condensadors polaritzats (o electrolítics) obliguen a que un dels borns sigui sempre el positiu i l'altre sempre el negatiu, per tant, solament poden funcionar amb c.c.

La importància dels condensadors polaritzats, malgrat el condicionant de la polaritat, sobre tot els de tàntal, consisteix en que s'aconsegueix una gran capacitat en un volum relativament petit en comparació amb els condensadors no polaritzats.



Diversos tipus de condensadors. El de la dreta és un condensador electrolític polaritzat, la resta són condensadors no polaritzats.

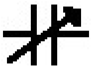
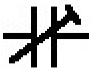


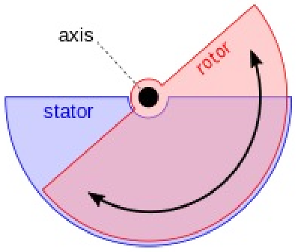


Black	0
Brown	1
Red	2
Orange	3
Yellow	4
Green	5
Blue	6
Violet	7
Grey	8
White	9

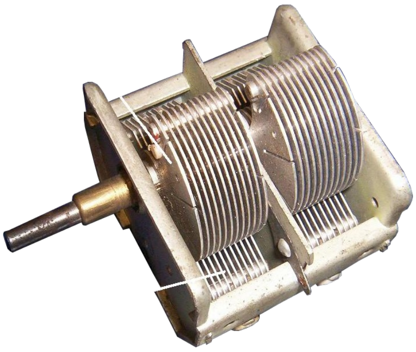
Els condensadors solen portar anotat sobre la seva càpsula el valor de la capacitat (habitualment en pF, a no ser que s'indiqui expressament la unitat), el voltatge màxim de treball i, en el cas de ser un condensador polaritzat, el born positiu i/o negatiu. Però, en alguns casos, enlloc de portar escrit el valor de la capacitat i tensió, pot ser que portin un codi de colors per indicar-ne els valors, en aquest cas, la taula lateral indica el codi de colors i la forma de interpretar-ho.

Condensadors variables

Els condensadors indicats prèviament són tots de valor fix, però, també existeix condensadors de valor variable. Aquests condensadors acostumen a ser de aire (és a dir, el material dielèctric que separa les dues plaques, o tàndem de plaques, és aire i acostumen a ser ajustables, o sigui, ajustables mitjançant un tornavís de plàstic per part dels tècnics o variables per part dels usuaris com és el cas dels condensadors de sintonia usats en els antics receptors de ràdio. El seu símbol habitual és:  o  en les seves variants de condensador variable o ajustable (trimmer) respectivament.



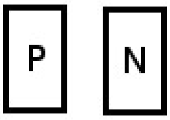
Condensador variable, superfície variable de les plaques enfrontades.



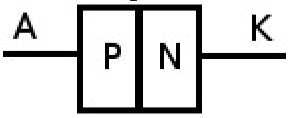
Doble condensador variable, utilitzat per la sintonia d'antics receptors de ràdio.

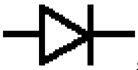
Condensador ajustable (trimmer)

3.5.- Diodes. Diodes LED.

Els semiconductors són materials formats per un monocristall, generalment de silici (Si) o germani (Ge), dopats amb substàncies adients (B, Al, Ga o P, As, Sb), per conferir-lis caràcter positiu (tipus P) o negatiu (tipus N), respectivament, 

.
Les diferents formes d'unió d'aquests monocristalls dopats són la base dels diferents dispositius semiconductors utilitzats en electrònica.

El diode és un component semiconductor, format per la unió de un semiconductor tipus P i un altre tipus N: 

. Té la particularitat que solament deixa passar el corrent des del costat ànode fins al càtode i en sentit contrari introdueix una oposició total al pas del corrent. El seu símbol convencional és: ,

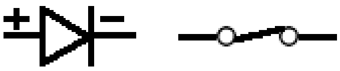
. En aquest símbol també es dedueix quin és el sentit del corrent en que és permet el pas i quin es el sentit en que es bloqueja.

Diode 1N4004. El càtode s'indica amb la franja blanca de l'esquerra. I_D=1A i V_R=400v

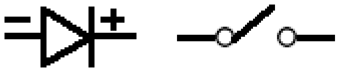
Diode 41HF120. I_D=40A i V_R=1200v

S'anomena **polarització directa** del diode quan aquest té aplicat el positiu al Ànode i el negatiu al Càtode i **polarització inversa** en cas contrari.

A efectes pràctics, podem considerar un diode com un interruptor controlat per la polaritat de la tensió entre els seus borns. En polarització directa es comporta com un interruptor tancat i en polarització inversa com un interruptor obert.



Polarització directa, equivalència a un interruptor tancat

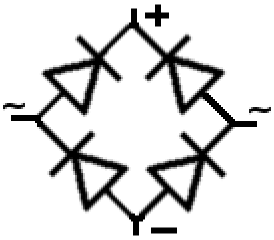


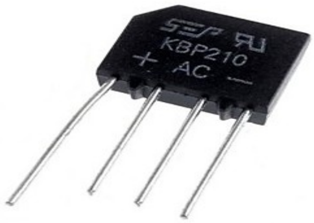
Polarització inversa, equivalència a un interruptor obert

En polarització directa el diode de silici té una tensió de creuament d'uns ~0'7v aproximadament. Aquest valor es pot menysprear en cas de tensions elevades, però, cal tenir-lo en compte quan les tensions són molt petites. S'entén per tensió de creuament la tensió mínima necessària per iniciar la conducció en polarització directa.

El diode semiconductor convencional descrit fins ara s'utilitza habitualment per controlar el sentit del corrent en el circuit electrònic i per fer tasques de rectificació (convertir c.a. en c.c.), per això, també se l'anomena diode rectificador. Aquest diode, segons el model i disseny, pot suportar tensions inverses de algunes desenes de volts fins a valors al voltant de algun miler de volts i intensitats directes de l'ordre d'alguns miliampers fins a alguns centenars d'ampers, com ara els models d'exemple: 1N4004 o el 41HF120 entre altres.

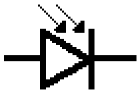
Aquests tipus de diodes es poden trobar individualment o també, formant el que es coneix com a pot de diodes o de Graetz, on quatre diodes estan interconnectats per formar un circuit rectificador de conversió de c.a. a c.c.



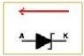
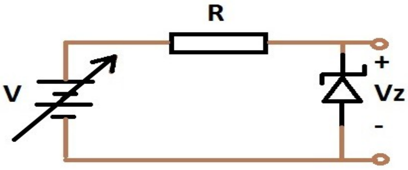


També existeixen altres variants de diodes que, mantenint les propietats inicials de solament permetre la conducció en polarització directa, també tenen altres propietats interessants i realitzen altres funcions. Entre altres, tenim els següents exemples:

Fotodiode
Aquest diode té la particularitat que quan està en polarització directa, no entra en conducció fins que no incideix sobre ell un feix de llum. Es a dir, és un diode sensible a la llum, si aquesta llum existeix i està en polarització directa, conduirà, sinó, no.



Diode zener
És un diode que treballa en polarització inversa i serveix per mantenir una tensió a un valor constant (tensió zener, V_z) encara que canviï la tensió que l'alimenta. És utilitzat per estabilitzar tensions que poden tenir fluctuacions dins d'uns determinats marges.

Diode túnel o Esaki
Aquest diode ofereix una conductància negativa en un tram de la seva corba característica. Se sol utilitzar en circuits d'amplificació/oscil·lació d'alta freqüència.

Diode varicap
Aquest diode està dissenyat per què en polarització inversa es comporti com si fos un condensador, amb la particularitat que la seva capacitat varia segons sigui el valor de la tensió inversa aplicada. Si la tensió inversa augmenta, disminuirà la capacitat i si la tensió inversa disminueix, augmentarà la capacitat. Actualment és molt utilitzat en els circuits de sintonització de receptors digitals ja que el valor de la capacitat és fàcilment ajustable segons la tensió aplicada i en conseqüència permet la sintonització d'emissores de radiofreqüència digitalment.

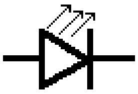
Diode Schottky

Aquest és un diode que admet freqüències de commutació molt més elevades que els diodes semiconductors convencionals amb unes pèrdues de commutació molt petites (baixa tensió de creuament ~0'25v). Això el fa útil en aplicacions de computació d'alta velocitat, encara que les tensions inverses i intensitat directa de treball que admet siguin de valors petits.

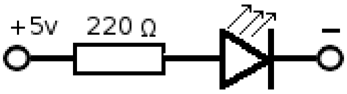
Diode emisor de llum (LED)

El LED (Light Emitter Diode) és un tipus de diode que té la particularitat que quan circula corrent en polarització directa a través d'ell, en la unió es genera una ona lluminosa d'una determinada longitud d'ona constant, que li confereix el color únic de il·luminació. En aquest cas, la llum ja es genera en el mateix color que es veu, per tant, no li cal cap filtre de color com passaria si la llum de color s'origina des d'una font de llum blanca. Cada color dels led's s'aconsegueix aplicant determinats dopatges en la unió semiconductora del diode. Per fer que un led normal s'il·lumini cal aplicar-li un corrent en polarització directa d'uns 20 mA aproximadament, superar excessivament aquest valor de intensitat pot destruir el led.

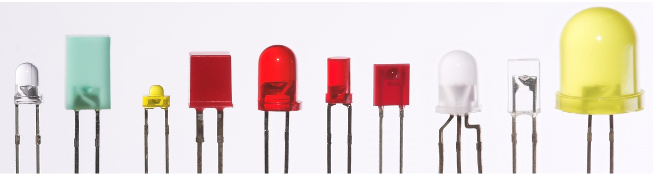
Per evitar la circulació d'un corrent excessiu a través del led cal connectar una resistència en sèrie que limiti aquest corrent. El valor de la resistència s'ha de calcular emprant la Llei d'Ohm, tenint en compte el valor de la tensió de creuament (~0'7v en cas de semiconductors de silici) i la tensió d'alimentació aplicada. Per exemple, si volem connectar un led a una tensió de 5v cal posar una resistència de: $R=\frac{5-0'7}{20\cdot10^{-3}}\approx220\Omega$, per qualsevol altra tensió on connectar el led, cal fer unes operacions similars per calcular la seva resistència en sèrie.

El símbol del Led és:  , l'ànode(+) i el càtode (-) s'identifiquen pel rebaix en el cercle que forma el led circular, el born més proper al costat recte correspon al càtode (-):

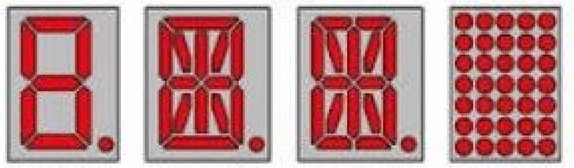
, també es poden identificar per la longitud de les potes (si no han estat tallades), la més llarga correspon al ànode (+).



Existeixen Led's de diversos colors: roig, groc, verd, blau, infraroig (iLed) etc. i de diverses formes: rodons de diferents diàmetres, rectangulars, segments, punts, etc. que es poden utilitzar per formar formes alfanumèriques i geomètriques.

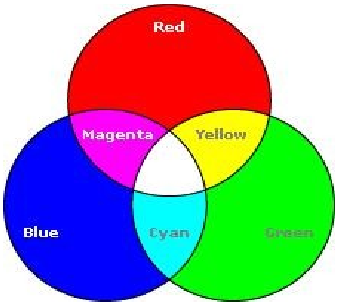


Múltiples colors i formes geomètriques de Led's.

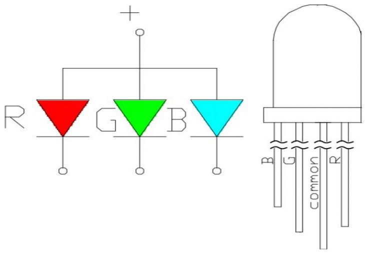


Display's Led de 7 segments, 14 segments, 16 segments i matriu de punts 7x5, respectivament.

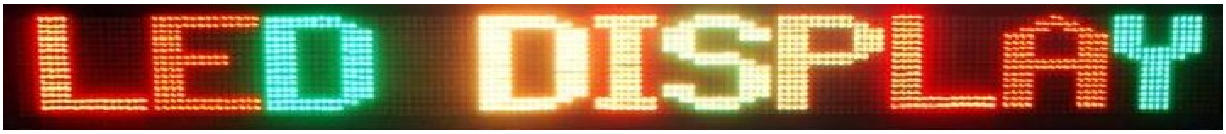
Led RGB
El Led RGB és una càpsula on s'ubica tres Led's conjuntament, de colors: roig, verd i blau.
Amb la mescla additiva de llum dels tres colors indicats, segons el percentatge de cadascun d'ells, s'aconsegueix formar tots els colors de l'espectre visible, inicialment, sense color (quan cap led està il·luminat, per absència de llum, fins al blanc, quan tots tres estan il·luminats amb la màxima intensitat). El connexionat d'aquest Led compost és igual com si es tractés de tres Leds individuals, encara que tenim un born comú a tots tres (càtode comú o ànode comú). Modificant el valor de la tensió de cadascun dels tres leds que formen el Led RGB, s'aconsegueix il·luminar-lo amb tots els colors de l'espectre visible.



Mescla additiva de colors per llum

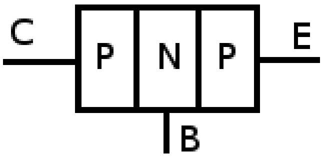


Led RGB d'ànode comú

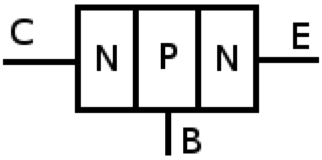


3.6.- Transistor bipolar i Mosfet.
El transistor (TRT) és el dispositiu electrònic que ens ofereix més possibilitats de control del corrent electrònic en el circuit. És un component semiconductor i n'existeix diversos tipus, entre ells, el més convencional: transistor bipolar (BJT), però, també hi ha altres tipus com ara els transistors FET, MosFET, IGBT, etc.

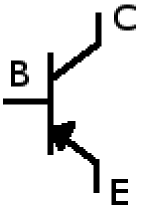
Transistor bipolar (BJT)
El transistor bipolar (BJT) està format per la unió de tres semiconductors tipus N i P. D'aquesta unió obtenim els dos tipus de transistors bipolars: el NPN i el PNP. Cadascuna d'aquestes pastilles semiconductores correspon a una de les potes de connexió del transistor que són: Col·lector (C), Emissor (E) i Base (B).



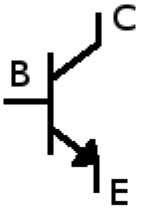
Unió PNP



Unió NPN

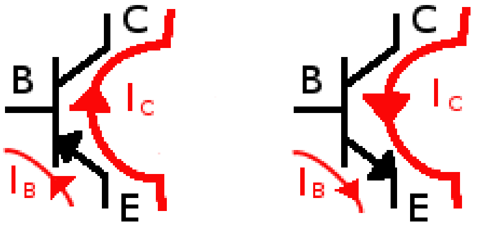


Símbol transistor PNP

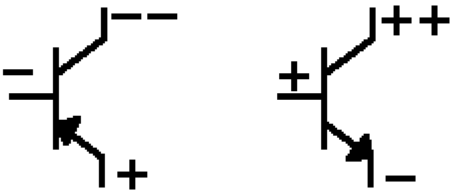


Símbol transistor NPN

Podem considerar al transistor bipolar com a un nus elèctric on conflueixen tres corrents: la corrent de base, la corrent de col·lector i la corrent d'emissor. Es compleix que: $I_E=I_B+I_C$. Els corrents que circulen són entre la base i el emissor i entre el col·lector i l'emissor, mai circularà corrent entre col·lector i base, a no ser que s'hagi espatllat el transistor. El sentit de la fletxa de l'emissor indica el sentit del corrent de la base i del col·lector. Per tant, en el transistor tindrem dos circuits: el circuit B-E i el circuit C-E, el primer circuit controla al segon, és a dir, amb el corrent entre B-E podem regular el valor del corrent entre C-E. El corrent de Base sempre és molt més petita que el corrent de Col·lector: $I_B <<< I_C$.



Sentit dels corrents de base i de col·lector per cada tipus de TRT. Notem com el corrent de Base és molt menor que el de Col·lector i a l'Emissor tenim la suma dels dos.



Polarització de les tensions a cada born del TRT per garantir els sentits i magnituds dels corrents. El doble signe ++ o - -, vol dir que és més positiu o més negatiu, respectivament.

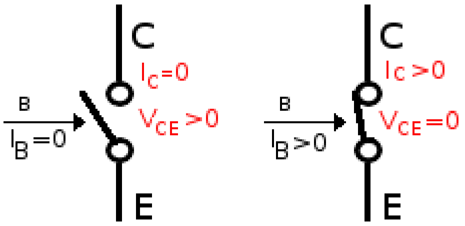
S'entén per polarització del transistor l'aplicació de les tensions amb els valors adequats als seus borns E, C i B per aconseguir que els corrents que hi circulin siguin amb els valors i sentits adequats. Les diferents tensions es poden obtenir des de fonts d'alimentació de valors diferents, o bé, el més habitual, utilitzar resistències del valor adequat que facin de divisors de tensió i facilitar les tensions i corrents en els valors adequats. Les resistències formen el circuit de polarització del TRT.

El transistor pot funcionar segons dos modes de funcionament: mode lineal o d'amplificació i mode de commutació o tall-saturació. El que el TRT treballi en un mode de funcionament o altre, depèn de quins siguin els valors de les tensions de polarització i corrents de base i col·lector aplicats i els valors de les seves corbes característiques.

Quan treballa en mode lineal, el corrent de col·lector depèn directament del valor del corrent de base i la relació és un coeficient β o h_{fe}, anomenat guany o factor d'amplificació. Aquest mode de funcionament és l'habitual en circuits amplificadors de senyal (àudio o RF) i tenim que $I_C = \beta \cdot I_B$.

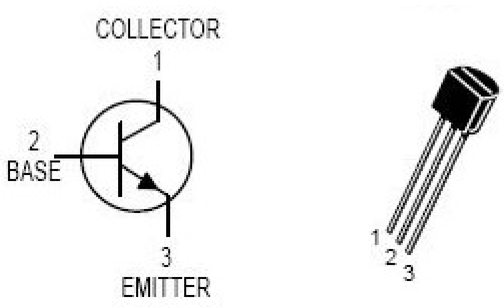
En canvi, quan el TRT treballa en mode de commutació, el corrent de col·lector també depèn del corrent de base, però, ara no hi ha una relació proporcional, sinó que el corrent de col·lector valdrà el màxim possible segons la càrrega que tingui connectada si el corrent de base és un valor prou elevat per arribar a la saturació i, valdrà 0 quan el corrent de base també valgui 0. En aquest mode

de funcionament considerarem al TRT com un interruptor, on la seva obertura i tancament depèn del valor del corrent de base.

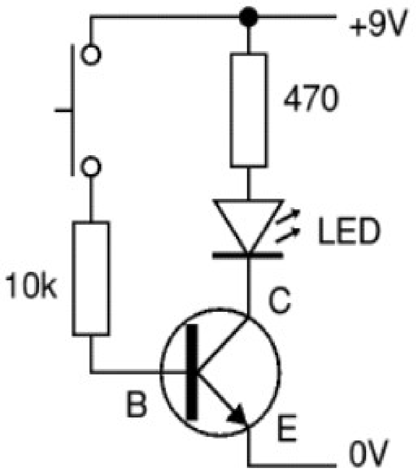


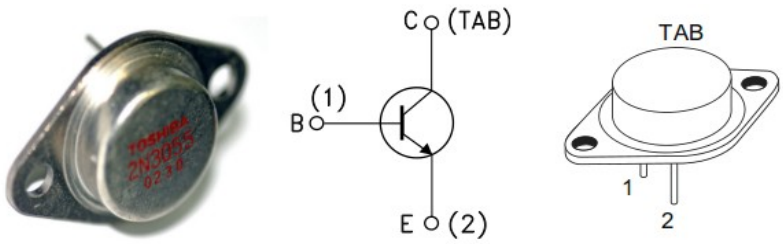
Comportament equivalent a un interruptor d'un TRT en mode de commutació

Els transistors emprats en circuits electrònics digitals, sempre treballen en mode commutació i el seu comportament equival al de un interruptor entre col·lector i emissor, controlat pel corrent de base.

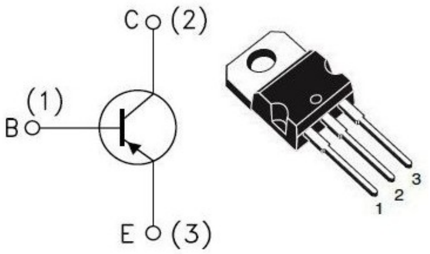


Transistor NPN BC547, per aplicacions d'amplificació i commutació. I_C=100mA, V_{CE}=45v, amb càpsula TO-92





Transistor NPN 2N3055, per aplicacions generals de commutació i amplificació. V_{CE}=60v, I_C=15A, amb càpsula TO-3



Transistor PNP TIP32, V_{CE}=40v, I_C=1A, amb càpsula TO-220

Transistor MOSFET

En el transistor BJT, es necessari un corrent de base per que existeixi el corrent de col·lector. En ocasions no ens interessa controlar el corrent C-E mitjançant un altre corrent de base.

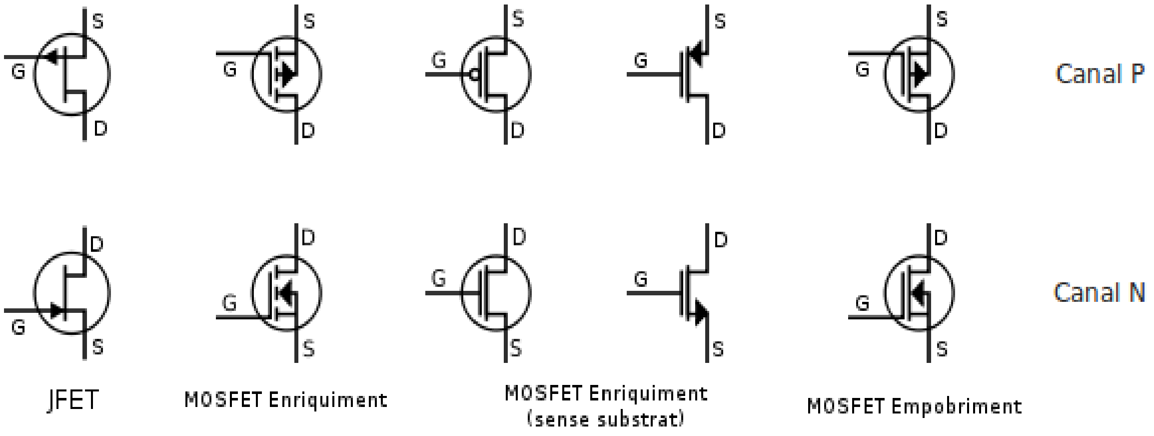
El transistor MosFet (Metal Oxid Semiconductor, Field Effect Transistor) és un transistor que controla el corrent entre Drenador (D) i Sortidor (S) mitjançant un camp elèctric aplicat a la Porta (G). Aquest és un transistor usat habitualment en circuits de potència o en circuits integrats i el

principal avantatge és que enlloc d'usar un corrent de base per controlar-lo, usem un camp elèctric que evita la interacció entre el control i el circuit de potència.
Per les seves característiques té un reduït consum en el circuit de porta (G) i això és òptim per a la seva integració en xips.

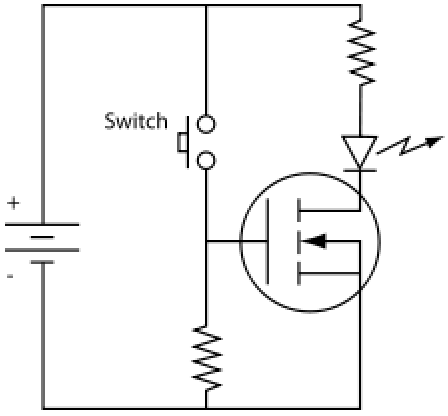
Existeixen els MosFet d'Enriquiment (en aplicar el camp elèctric es tanca el canal i condueixen) i els MosFet d'Empobriment (en aplicar el camp elèctric s'obre el canal i es talla la conducció), a la vegada que també poden ser de tipus P o N, segons sigui el tipus de semiconductor del substrat.

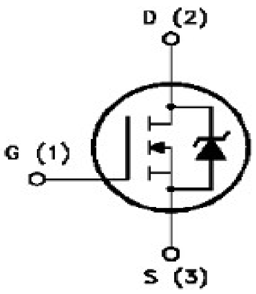
A efectes pràctics, funcionant en commutació, el podrem considerar també com un equivalent a un interruptor entre D i S, controlat pel camp elèctric de la G.

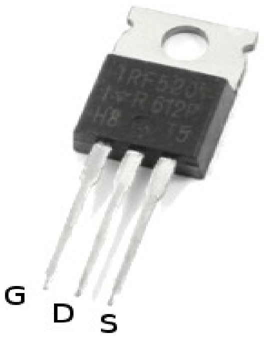
Tipus de transistors MosFet (inclou també, transistors Jfet):



Exemple d'aplicació d'un MosFet en un circuit per il·luminar un Led:



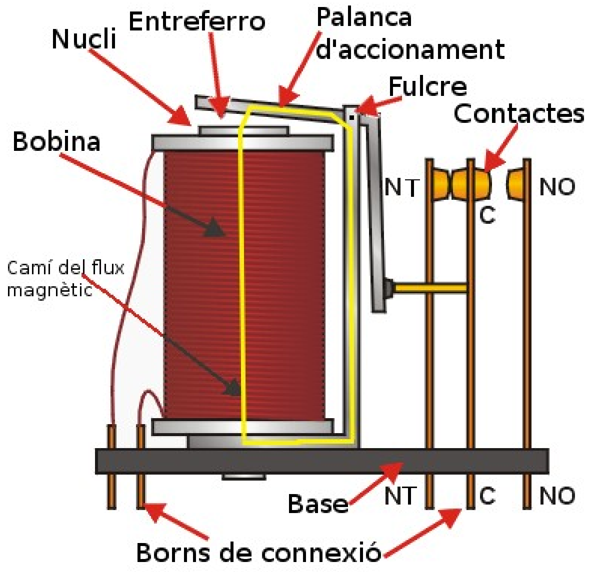




Transistor MOSFET Canal N d'enriquiment, IRF520, per aplicacions de potència: Control motors, Amplificació d'àudio, Convertidors CC-CC i CC-CA, Reguladors, Acció de bobines i relés, etc.. V_{DSS}=100v, I_D=10A, amb càpsula TO-220

3.7.- Relés.

Un relé és un mecanisme electromecànic que permet canviar l'estat d'un interruptor o commutador per l'acció d'un camp electromagnètic originat per una bobina.



Relé electromagnètic d'un circuit

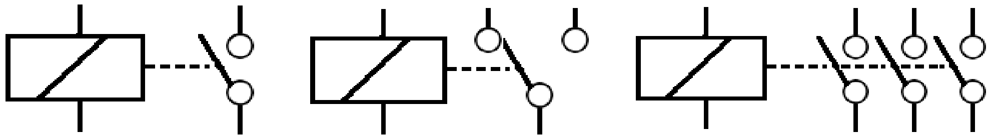
En el mateix accionament electromecànic, es pot actuar sobre múltiples contactes NO, NT i commutadors. Cada commutador rep el nom de circuit, així podem tenir relés d'un circuit, dos circuits, tres, etc.

Funcionament: La figura ens mostra la posició de repòs del relé, on es veu com el contacte C està en connexió amb NT. En aplicar una tensió en els bors de connexió de la bobina, aquesta origina un camp magnètic que es tanca a través del camí del flux magnètic indicat. En el entreferro apareix un pol Nord i un pol Sud que produeixen l'atracció magnètica de la palanca d'accionament que pivota sobre el fulcre, en aquest moviment, modifica la posició dels contactes fent que C connecti ara amb el NO. Quan desapareix la tensió de la bobina, desapareix el camp magnètic i la palanca d'accionament retorna a la seva posició de repòs fent que el contacte C torni a connexionar amb NT. Els contactes NT i NO s'utilitzen per connectar un altre circuit comandat pel relé.

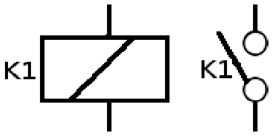
Quan el relé s'utilitza per comandar potències elevades, per exemple, engegada de motors elèctrics, control de il·luminació de potència, etc. cal tenir uns contactes elèctrics i un accionament electromecànic preparat per suportar aquestes potències elèctriques més elevades. En aquest cas, al relé se'l coneix amb el nom de Contactor, tenint un principi de funcionament igual que el descrit però, per tensions i corrents elevats.

El relé té múltiples aplicacions en el camp de l'automatització elèctrica ja que es la base dels automatismes cablejats. Amb ell s'implementen automatismes complexes que poden contenir desenes de relés per implementar una funcionalitat lògica.
Una altra utilitat molt important d'un relé consisteix en la separació de circuits, és a dir, en ocasions tenim sistemes de control que treballen a petites tensions i corrents, com es el cas de circuits electrònics de control, i necessitem controlar altres elements que treballen amb tensions i corrents més elevats, llavors, el relé, en aquest cas, ens fa la funció de separar els circuits elèctrics, mantenint la relació de control desitjada.

El símbol del relé és el següent, on la bobina es representa pel rectangle amb una línia travessera. Hi podem veure el cas accionar un o més contactes simultàniament:



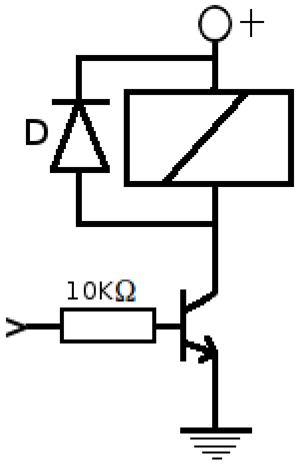
Quan en un esquema elèctric o electrònic la bobina no es pot representar prop del contacte que acciona per unir-los amb una línia de punts, se sol representar les dos parts amb la mateixa referència:



En electrònica utilitzem els relés per accionar circuits que treballen a tensions i/o corrents més elevades, com ara encendre i apagar llums que funcionen a 220v, accionar motors, etc.

Protecció de transistors amb el "diode volant".

La bobina del relé és un element inductiu. Les bobines tenen la particularitat que quan es talla el corrent que circula per ella, reacciona provocant una sobre tensió per forçar la continuació de la circulació del corrent a través de la mateixa bobina i extingir-la lentament (és a dir, la descàrrega de la bobina). Aquest fenomen electromagnètic no és cap problema greu quan la bobina és accionada per un interruptor mecànic, però, no podem dir el mateix en el cas de ser accionada per un transistor, tal com es el cas habitual en circuits electrònics.



Quan una bobina es accionada per un transistor en mode commutació i passa a l'estat de tall, la descàrrega de l'energia acumulada a la bobina provocaria l'existència d'una sobretensió entre els borns Col·lector i Emissor del TRT que el destruiria. Per evitar això cal posar un diode D en paral·lel amb la bobina, de tal manera que el diode serà el que tancarà el circuit del corrent de descàrrega de la bobina, evitant la sobretensió i protegint al transistor, tal com es veu a l'esquema lateral. Cal assegurar-se que el diode està connectat en el sentit indicat, sinó no farà cap efecte. El diode D, ubicat en paral·lel amb la bobina del relé per protegir el transistor s'anomena "diode d'efecte volant" o, simplement, "diode volant", de l'anglès "Free wheeling diode", en comparació amb els volants d'inèrcia de màquines, que absorbeixen i cedeixen energia mecànica.

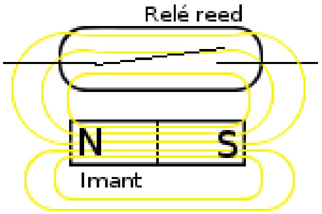
Aquest concepte de protecció dels transistors mitjançant un diode, és extensible a qualsevol altre dispositiu que contingui una bobina accionada per un transistor, com ara, motors, electrovàlvules, etc.

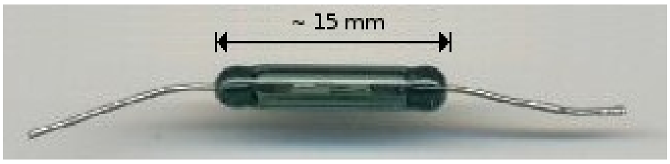
Altres tipus de relés.

El relé descrit fins ara és el relé electromecànic o convencional, però, existeixen altres dispositius, també anomenats relés que tenen una utilitat i funcionament diferents. A continuació es descriu alguns d'ells, tot i que encara n'existeix altres més amb diferents funcionalitats més específiques.

Relé reed o de làmines

És un, o més, contactes elèctrics, normalment oberts, ubicats dins una petita ampolla de vidre on hi ha el buit o un gas inert. Quan s'apropa un camp magnètic (d'un imant permanent o electroimant) al seu entorn, els contactes es tanquen i quan desapareix el camp magnètic es tornen a obrir.
Aquest tipus de relé és de reduïdes dimensions i se sol utilitzar com a detector de la posició d'algun altre dispositiu que contingui l'imant activador (permanent o electroimant).





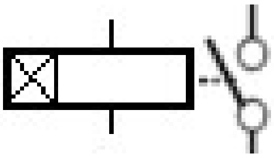
Contacte reed. Quan s'apropa un camp magnètic, el contacte de dins l'ampolla es tanca.

Relé d'estat sòlid

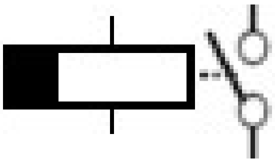
El relé electromecànic té l'inconvenient del seu moviment mecànic que produeix sorolls, en limita la vida útil i no en permet cicles de treball, obertura i tancament per unitat de temps, massa elevats. A més, en la connexió i desconnexió dels seus contactes elèctrics és poden produir guspires. Aquestes característiques fan que no sigui adequat per alguns tipus d'aplicació especial, per exemple, treball en zones de risc d'explosió, llocs on sigui necessari el silenci o maniobres que requereixen elevades freqüències del cicle de treball.
El relé d'estat sòlid pretén evitar aquests inconvenients. No conté ni bobina ni contactes mecànics, tot és estàtic. Està format únicament per components electrònics i la connexió i desconnexió del circuit de potència s'efectua per uns components anomenats tiristors i/o triacs.

Relé temporitzador

N'hi ha de diversos tipus, encara que els modes més representatius són el temporitzador a la connexió i el temporitzador a la desconnexió. Habitualment és pot regular els temps de temporització. Aquest tipus de relé disposa d'un sistema retardador de la connexió o de la desconnexió dels contactes a partir del moment en que s'aplica tensió a la bobina, o es treu la tensió a la bobina, respectivament. El retard de connexió o desconnexió s'aconsegueix, habitualment per mitjans pneumàtics o electrònics.
Els símbols representatius d'aquests relés temporitzats són:



Relé temporitzat a la connexió. L'activació del contacte és fa un temps més tard després d'haver activat la bobina.



Relé temporitzat a la desconnexió. La desactivació del contacte és fa un temps més tard després d'haver desactivat la bobina.

Aquests tipus de relés es poden trobar comercialment de forma independent o bé, tots ells inclosos en un mateix dispositiu on és pot programar els diferents modes de funcionament. A part dels modes de funcionament indicats en pot haver altres més.

Relé pas a pas o telerruptor

A diferència del relé convencional que solament té un estat estable, el telerruptor té 2 o més estats estables. És a dir, quan s'aplica tensió a la bobina es connecten els contactes i així queden, encara que s'elimini la tensió a la bobina. La propera vegada que s'aplica tensió a la bobina es desconnecten els contactes i també queden així, encara que es desconnecti la tensió a la bobina. És a dir, l'activació/desactivació dels contactes es produeix a cada impuls de tensió que s'aplica a la bobina.

Relé tèrmic

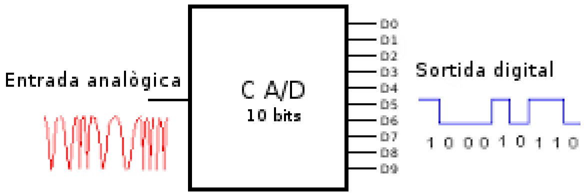
Es coneix amb el nom de relé tèrmic aquell que desconnecta el circuit quan s'està produint un elevat consum de corrent elèctric que produeix una elevada dissipació tèrmica. Aquest tipus de relé s'utilitza generalment amb la protecció de motors elèctrics de potència i té com a funció detectar sobrecàrregues, per mitjà del calor produït pel pas del corrent al voltant de uns detectors tèrmics bimetalls.

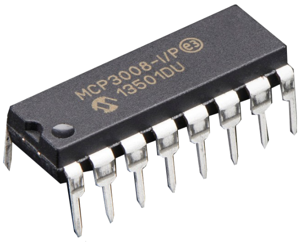
3.8.- Circuits convertidors A/D i D/A.

Les dades i informació del nostre entorn es troba, generalment, en forma analògica, però, el tractament que se'n fa de la informació per part dels sistemes electrònics és en forma digital. Per tant, en moltes ocasions és necessari adquirir les dades del nostre entorn en forma analògica, convertir-la a digital, fer-ne el tractament per part del circuit electrònic en format digital i, finalment tornar a convertir a analògic per subministrar, al nostre entorn, la informació tractada en el format analògic.

Entre altres dades, una característica que defineix la qualitat del senyal convertit es determina segons el nombre de bits digitals del convertidor. Per tant com més gran sigui el nombre de bits del convertidor millor precisió en la conversió, tot i que això implica una major complexitat en el circuit. Llavors considerarem normals convertidors A/D de 10 bits (1024 valors diferents) i convertidors D/A de 8 bits (256 valors diferents), encara que en el mercat se'n pot trobar de altres nivells de precisió.

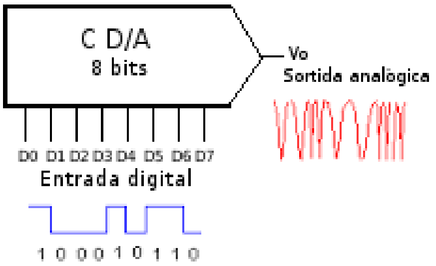
Convertidor analògic – digital:

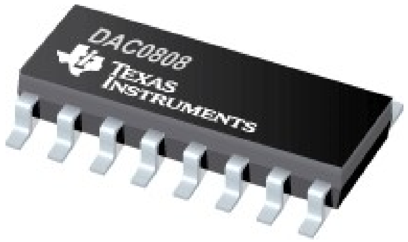




Xip MCP3008, C A/D de 10 bits i 8 canals, amb càpsula DIP16

Convertidor digital – analògic:





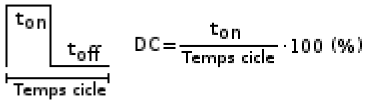
Xip DAC0808, C D/A de 8 bits, amb càpsula SOP16

3.9.- Modulació per amplada de pols, PWM.

La modulació PWM (Pulse Width Modulation), en un senyal que es va repetint periòdicament, consisteix en controlar el temps en que el senyal es troba a nivell alt, respecte del temps total del període o cicle. Si la freqüència és elevada (~500Hz), no s'observaran les variacions dels senyal i el resultat que s'obté és la mitjana de la tensió en un període.

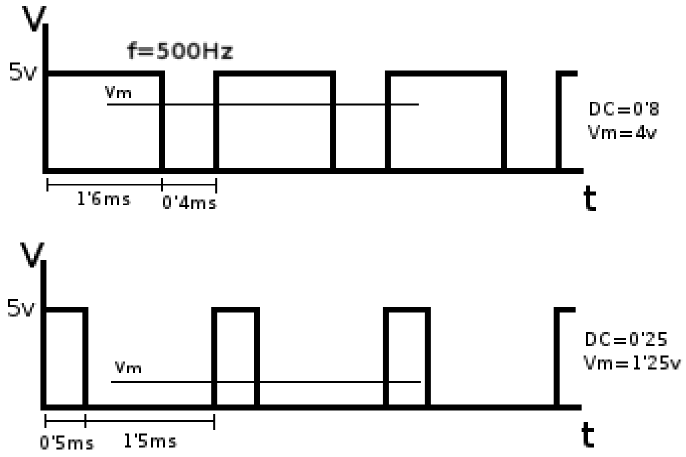
Aquesta tècnica substitueix en la majoria dels casos al CD/A i és emprada en aplicacions on no es requereix la precisió instantània del valor analògic convertit per part d'un convertidor digital/analògic (CD/A). És a dir, no serà útil si el que volem convertir de digital a analògic és un so d'àudio de qualitat, en canvi, si que ens és útil pel cas de control analògic de circuits electrònics com la velocitat de gir d'un motor, la il·luminació d'un Led, etc. i en molts casos, ens estalvia la complexitat d'implementar un CD/A.

En una modulació PWM, entenem per cicle de treball (Duty Cycle, DC) la relació percentual que existeix entre el temps que el senyal està a nivell alt i el temps total de durada del cicle.



La tensió mitjana vindrà determinada segons la tensió màxima i el DC: $Vm = V \cdot DC$

Exemple de tensions modulades amb PWM:



En comparació amb els CD/A, on la conversió s'efectua segons el nombre de bits del convertidor, l'escala de temps es pot substituir pel seu equivalent en el nombre de bits del convertidor. Així, per exemple, en el cas d'un convertidor CD/A de 8 bits tenim que existeix un nombre total de 2^8=256 valors (de 0 a 255), llavors tindrem que el nombre màxim de temps de cicle correspondria a 255 i si considerem, per exemple, una tensió de 5v i una conversió del valor numèric digital 140, aquest correspondria una tensió mitja de:

$$DC=\frac{140}{256}=0'5469\,;Vm=5\cdot 0'5469=2'73\,v$$

3.10.- Motor c.c., motor pas a pas i servomotor.

Els motors elèctrics són dispositius electromecànics que basen el seu funcionament en els fenòmens electromagnètics que es produeixen en les bobines i/o imants del seu rotor i estator. La seva funció consisteix en subministrar energia mecànica rotativa en el seu eix a partir de l'energia elèctrica que li apliquem.

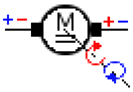
Existeixen múltiples tipus de motors elèctrics. La descripció de tots aquests i el seu funcionament detallat seria molt extens i no es l'objectiu d'aquest dossier, però, si que es comentarà les principals característiques bàsiques dels motors coneguts amb el nom de: "motor de corrent continu", "motor pas a pas", "servomotor" i els seus accionaments per mitjans electrònics.

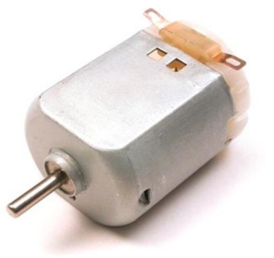
Motor de corrent continu

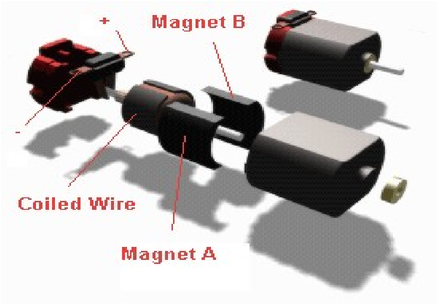
El motor de c.c. funciona solament amb c.c., tal com indica el seu nom, i la seva velocitat de gir depèn, i és directament proporcional, al valor de la tensió que se li apliqui. Per aconseguir la inversió de gir només cal invertir la polaritat de la tensió aplicada.

El de petites potències sol està format per un estator d'imants permanents i un rotor que conté una bobina, alimentada per la tensió de c.c. aplicada al motor mitjançant un col·lector de lamel·les i unes escombretes.

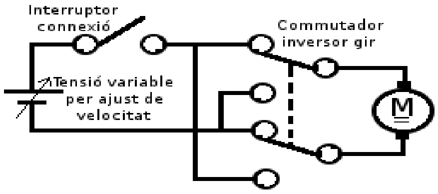
El símbol del motor de c.c. és:

Invertint la polaritat aconseguim la inversió de gir: 





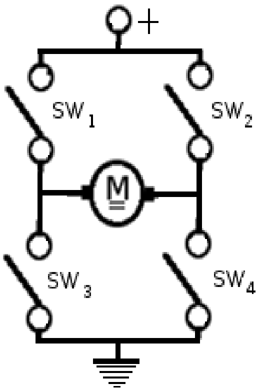
Per controlar la connexió i desconnexió d'un motor c.c., el seu sentit de gir i velocitat, podem utilitzar contactes elèctrics i generadors de tensió variable, formant un circuit similar al següent:



Però, aquest circuit és molt rudimentari, ja que el doble commutador ha de ser d'accionament manual o bé, mitjançant un relé i la font d'alimentació variable és una mica més complex d'obtenir. Per sort, disposem de circuits electrònics específicament dissenyats per controlar els motors de c.c. de forma més àgil. Entre altres tenim el circuit pont en H i els circuits integrats controladors de motors.

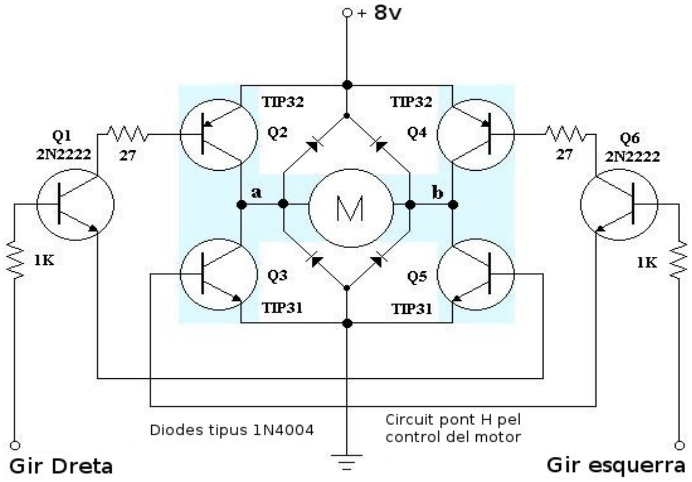
Pont H

Aquest circuit correspon a una combinació de transistors i el motor a controlar que formen un pont en H.



La imatge lateral representa una simplificació amb interruptors d'un pont en H. Per fer que el motor giri cap a la dreta haurem de connectar simultàniament els interruptors SW1 i SW4, per a fer que el motor giri cap a l'esquerra haurem de connectar els interruptors SW2 i SW3. En aquest circuit s'ha de vigilar de no connectar simultàniament el SW1 i SW3 o el SW2 i SW4, ja que es produiria un curtcircuit.

Per regular la velocitat de gir del motor es podria fer mitjançant l'obertura i tancament molt ràpid dels interruptors corresponents al gir desitjat, de tal manera que introduiríem un cicle de treball com l'explicat a l'apartat de modulació PWM, però, això no pot ser ja que els interruptors mecànics no admeten una obertura i tancament tan ràpids com els necessaris per aconseguir la regulació de velocitat desitjada, per això, els ponts H reals s'implementen amb transistors (tipus BJT, MosFet, IGBT, etc.) que treballen en mode commutació i permeten unes freqüències de tall i conducció prou elevades com per admetre la regulació PWM de la velocitat.

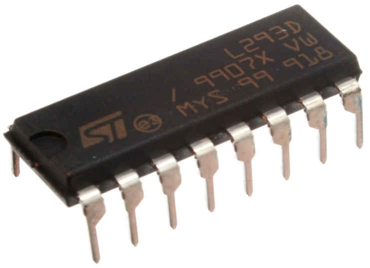


El circuit lateral correspon a un pont H real implementat amb transistors BJT, els transistors i motor que corresponen al pont H són els que es troben marcats en la franja de color blau, la resta són elements auxiliars de control i protecció.

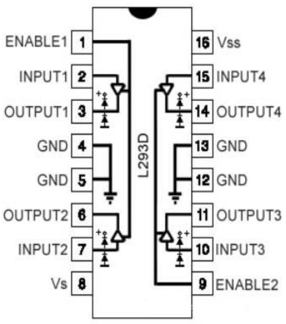
Els quatre diodes són per protegir als transistors de les corrents de descàrrega de la bobina del motor que es produeix en les connexions i desconnexions, són els diodes "volants" explicats en el mateix cas en l'apartat de Relés.

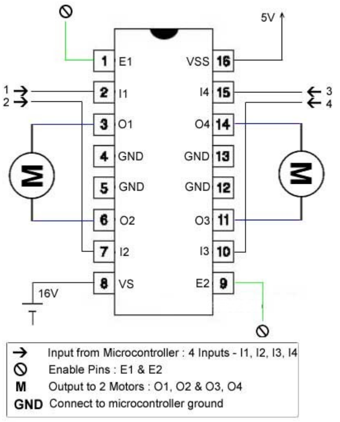
L'activació s'ha d'efectuar simultàniament en els transistors de cada parella Q2 i Q5 o Q4 i Q3, veiem que aquestes parelles són: PNP i NPN permeten el pas del corrent en el mateix sentit, de + a -, pels circuits Col·lector i Emissor de cada TRT a través del motor.
L'activació conjunta dels dos TRT de cada parella s'aconsegueix amb els transistors auxiliars Q1 i Q6, corresponents a cada parella, provocant l'existència, o no, simultàniament, dels corrents de base als dos TRT, provocant el seu pas a conducció o tall, segons correspongui.
Com que els TRT poden commutar a freqüències molt elevades, en aquest cas es possible aplicar senyals PWM als borns de "Gir dreta" o "Gir esquerra" i ajustar la velocitat de gir al nivell desitjat.

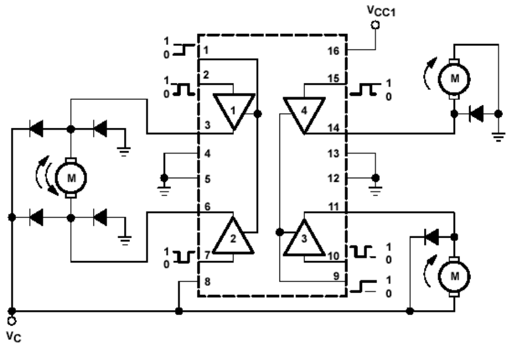
Implementació amb C.I.
El pont H descrit prèviament es pot trobar implementat en un circuit integrat, això serà útil sempre que les potències i tensions que vulguem utilitzar s'adaptin a les característiques del C.I. Existeixen diversos xips controladors de motors, com ara el L298N, LMD18200T, L6203, etc. Però, aquí veurem la implementació mitjançant el I.C. L293D. Aquest circuit integrat inclou dos ponts H, per tant, permet controlar dos motors simultàniament en el doble sentit de gir. El circuit s'alimenta a 5v pel pin 16, però, la tensió de funcionament dels motors pot ser més elevada i s'aplica pel pin 8. Controlarem el motor de la dreta aplicant una tensió pel pin 15 o 10 segons vulguem que giri en un sentit o altre, si la tensió aplicada en aquests pins és una senyal PWM podrem regular la velocitat de gir. Pel motor de l'esquerra en controlem el sentit de gir pels pins 2 i 7.



El circuit permet altres configuracions en cas que no vulguem tenir el doble sentit de gir, en aquest cas podríem controlar fins a quatre motors en un sol sentit. Els diagrames següents mostren el diagrama intern del xip i les connexions a fer per controlar a dos motors en els dos sentits de gir, o tres motors, dos en un sol sentit i un en dos sentits de gir.







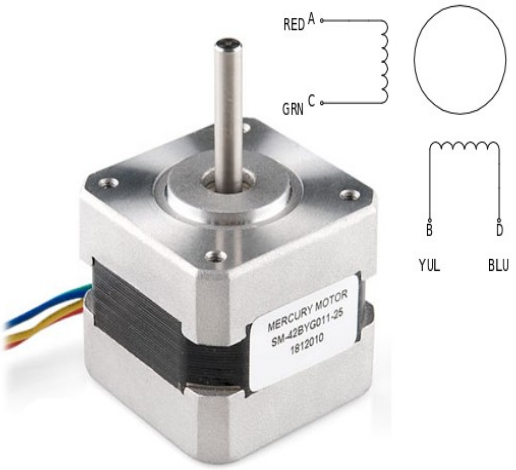
Els motors connectats al L293D no requereixen diodes volants de protecció ja que aquests es troben implementats dins el mateix I.C. No es pot dir el mateix del model equivalent L293.

Motor pas a pas
El motor pas a pas (stepper) és un motor que es mou un pas per a cada impuls rebut. És un tipus de motor de precisió que pot girar a dreta i esquerra amb diferents velocitats fàcilment regulables amb els impulsos i es pot aturar, amb retenció, a cada posició. Per tant, pot ser motor de gir continu i/o posicionar-se a determinats angles. El valor de cada pas sol ser d'entre 1'8º i 90º aproximadament, segons sigui el model de motor.

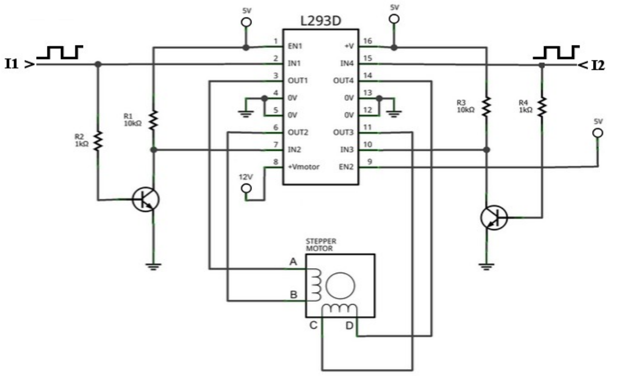
Consisteix en un rotor format per un imant permanent i un estator que conté múltiples bobines que s'activen controladament, generant els camps electromagnètics i pols N i S, amb la seqüència de passos adequada per fer girar, per l'atracció-repulsió magnètica de pols N i S, l'imant permanent del rotor amb l'angle i velocitat corresponents.

El seu control sempre s'efectua electrònicament.

Existeix dos tipus de motors PaP: els bipolars i els unipolars.



Motor PaP bipolar, SM-42BYG011-25, 12V, 330mA, 1'8º per pas, 200 passos per 360º, Γ=2'3Kg·cm



Esquema de control d'un Motor PaP bipolar amb L293D. Apliquem els impulsos per I1 i I2 seqüencialment des de les sortides digitals del sistema de control per formar els passos de gir, sentit, velocitat i aturada corresponents.

Servomotor

Un servomotor (també anomenat servo) es un dispositiu compost. Està format de diversos elements en una mateixa caixa. Concretament està compost de:

- Motor de c.c.
- Reductor mecànic amb un tren d'engranatges per reduir la velocitat de gir.
- Sensor de posició angular i/o velocitat (generalment un potenciòmetre, encara que pot ser algun altre sistema, com ara els encoders òptics)
- Circuit electrònic que llegeix l'entrada de consigna i la posició de l'eix, i determina, per comparació, el moviment a efectuar per part del motor fins posicionar l'eix al valor de consigna.

Tot el conjunt forma un servosistema realimentat, on comparem l'angle de posició de gir de l'eix de sortida i/o la velocitat, amb el valor indicat en la consigna introduïda pel senyal. El servomotor pot regular tan la seva velocitat i sentit de gir com també la seva aturada estable en un angle determinat pel propi control. Els servos convencionals usats en RC (radiocontrol) poden girar 180º, aproximadament, encara que alguns també giren fins a 270º. Altres models de servos poden ser també multivolta (servos de rotació contínua 360º) i actuar com a motor convencional amb reductor, ajustant la seva velocitat i sentit de gir i el nombre de graus girats segons el valor del senyal PWM que li apliquem.

Servo model SM-S2309S. Angle rotació ±60º. A 6v: Γ=1'3kg·cm, v=0'09s per 60º. Connexions: vermell, negre:+ i -, blanc: senyal PWM de control.

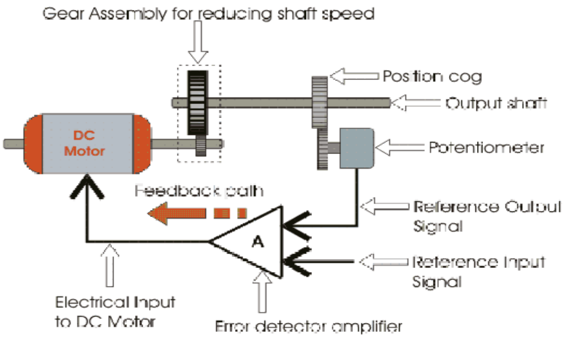


Diagrama en blocs d'un servomotor. Sistema de llaç tancat amb la realimentació de la posició angular de sortida que es compara amb el valor de consigna de l'angle indicat en el senyal de control, per determinar l'error (diferència) i actuar sobre el motor.

Parts components del servo. S'observa el motor de c.c. convencional, els engranatges reductors, el potenciòmetre sensor de posició angular i l'electrònica de control del conjunt.

Servo model SM-S4303R. Rotació continua 360º. A 6v: Γ=2'8kg·cm, v= 0'14s per 60º ~ 71rpm. Connexions: vermell, negre:+ i -, blanc: senyal PWM de control.

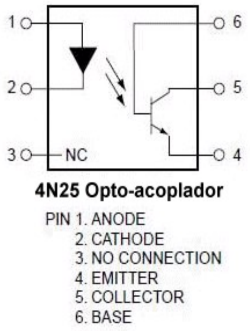
3.11.- Optoacobladors.

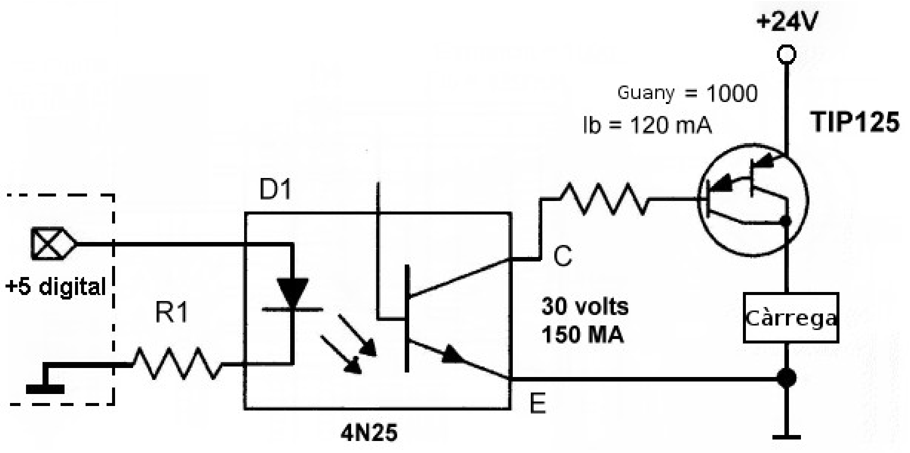
Els optoacobladors són uns circuits integrats compostos[1] generalment per un led i un fototransistor[2]. Quan el led s'il·lumina, el fototransistor entra en estat de conducció, comportant-se com un interruptor tancat. Si el led no està il·luminat, el fototransistor estarà en estat de tall.

La seva utilitat consisteix en la separació de circuits, on l'únic nexe que hi haurà serà l'acoblament òptic. Així podem controlar un circuit electrònic que treballa a altes tensions i potències mitjançant un circuit de petita tensió com és l'habitual en circuits de control, sense perill que com a conseqüència d'una avaria, les tensions elevades puguin afectar al circuit de control.

1 La composició pot estar formada per diferents elements òptics: led-fotoresistència, led-fotodiode, led-diac, doble led en antiparal·lel-transistor, led-darlington, etc., sempre tancats dins la mateixa caixa per tenir el millor acoblament òptic.
2 El fototransistor és com un transistor bipolar normal, però, la conducció col·lector-emissor també és pot iniciar aplicant llum a la unió base-emissor, a part de la habitual corrent de base.

Les següents imatges corresponen a l'optoacoblador 4N25 i un exemple d'aplicació per controlar a una càrrega que treballa a una intensitat i tensió molt més elevada que la de control.



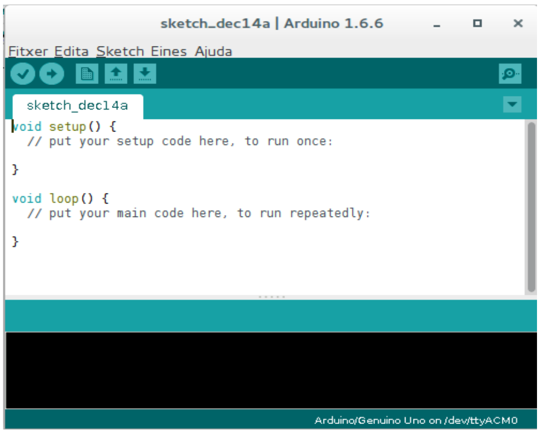


4.- Programació en C.

Arduino implementa una part adaptada del llenguatge de programació C per al desenvolupament dels programes de control. Podem trobar una descripció extensa d'aquest llenguatge de programació, derivat de Wiring, adaptat a Arduino a la web: https://www.arduino.cc/en/Reference/HomePage.

En aquest apartat farem una introducció bàsica dels conceptes i tècniques de programació necessaris per les pràctiques implementades posteriorment, però, pels detalls més específics caldrà visitar la web de referència indicada prèviament.

Arduino proveeix d'un IDE per desenvolupar els programes que es pot obtenir des de la web: https://www.arduino.cc/en/Main/Software, aquest IDE proveeix tots elements necessaris per desenvolupar els programes i carregar-los al circuit. És el que es pot veure a continuació:



A la part central del IDE és on escrivim el programa, per defecte s'hi crea les dues funcions bàsiques que no han de faltar en cap programa: *setup()* i *loop()*. La part inferior és una finestra de comunicació on apareixeran els missatges del compilador, la transmissió i els errors. Al peu podem veure el port sèrie de comunicació amb la placa (a la imatge: /dev/ttyACM0). Els següents botons del IDE tenen la funcionalitat següent:

Verifica la sintaxis del programa.

Compila el programa i el transmet a la placa Arduino.

Crea un nou programa buit.

Obre un programa des d'un fitxer .ino

Desa el programa a un fitxer .ino

Obertura de la finestra de comunicació sèrie.

La finestra de comunicació sèrie ens permet comunicar-nos amb el programa creat, des del mateix ordinador. És útil per enviar i rebre missatges que ens permeten veure i modificar l'evolució del programa.

4.1.- Estructura bàsica dels programes per Arduino.

Els programes per Arduino consten de dos funcions bàsiques: **setup()** i **loop()**. En la primera posarem el codi que s'haurà d'executar solament en el moment d'iniciar el programa i en la segona, posarem aquelles instruccions que s'hauran de repetir constantment en un bucle sense fi.

```
void setup() {
  // introduïm aquí el codi que s'ha d'executar una sola vegada al inici.
}

void loop() {
  // introduïm aquí el codi principal que es repetirà indefinidament.
}
```

Totes les instruccions del programa han d'acabar amb un punt i coma: **;** (excepte directives i blocs d'instruccions).
Podem introduir comentaris utilitzant: **//** si volem que siguin en una sola línia o bé: **/* */** si volem que el comentari ocupi més d'una línia.
Els blocs d'instruccions s'encabiran entre claus: **{ }**
Els arrays indicaran els seus ítems entre claudàtors: **[ ]**

4.2.- Llibreries.

Per defecte, el llenguatge interpreta les instruccions bàsiques, suficients per als circuits més bàsics, però, quan cal utilitzar algun perifèric específic, cal indicar-ho expressament per que es carreguin les funcions pròpies del perifèric que es vol implementar.

Les llibreries s'inclouran amb: `#include <nomLlibreria.h>` al principi del programa, abans de la funció setup(). Existeixen múltiples llibreries per fer tractament específics, alguns exemples són:

`#include <LiquidCrystal.h>` Per incloure les funcions específiques del display LCD.
`#include <Servo.h>` Per incloure les funcions específiques de control del Servomotor.
`#include <SD.h>` Per incloure les funcions específiques de la tarja microSD.

L'IDE d'Arduino permet inserir automàticament les llibreries segons el dispositiu a controlar, accedint al menú: Sketch / Include Library i seleccionant el dispositiu addicional a gestionar.

4.3.- Tipus de dades. Constants i variables. Operadors.

Els tipus de dades que interpreta el programa són les void, char, int, long, float, string, array, etc. entre altres.

void significa un tipus buit o inexistent, s'utilitza com a indicador del tipus de retorn de les funcions, quan aquestes no retornen cap valor.
char és un tipus de dades adequat per encabir una sola lletra (en un byte).
int és un tipus de dades numèriques enters entre -32768 i 32767
long és un tipus de dades numèriques enters entre -2,147,483,648 i 2,147,483,647
float és un tipus de dades numèriques de coma flotant entre $3.4028235x10^{+38}$ i $-3.4028235x10^{+38}$. És el tipus que ens permet operar amb valor numèrics amb decimals: 12.0, 14.33, etc.
string és un tipus de dades equivalent a un array de char (`char nom[25];`) per encabir dades alfanumèriques.
String és un objecte per utilitzar i manipular cadenes de caracters (alfanumèriques).
array és una col·lecció de variables de mateix tipus, amb un nom comú i indexades.

Existeixen més tipus de dades que es troben descrites a la web d'Arduino, a més d'una explicació més extensa de les que aquí s'han presentat.

Les **constants** són aquelles dades que no canviaran en l'execució del programa. Es declaren mitjançant la directiva `#define nomConstant valor` (sense ; final) o la instrucció:
const (Exemple. `const float PI = 3.14;`).

Hi ha constants que ja han estat predefinides pel mateix llenguatge, algunes d'aquestes són: **HIGH**, **LOW**, **true**, **false**, **INPUT**, **OUTPUT**, etc. És una costum habitual escriure el nom de les constants en majúscules.

Les **variables** són zones de memòria, que tenen un nom assignat i un tipus de dades, en elles s'hi emmagatzema de forma temporal dades en el format que tinguin definits en la declaració. És necessari declarar totes les variables prèviament, o en el mateix moment, a la seva primera utilització. Els noms de les variables no poden començar per números ni contenir caràcters especials ni espais en blanc. Exemples de definició de variables (algunes amb assignació inclosa):

```
int nombreFiles;
float valMajor=12.3;
char nom[]="Joan";
String nom = "Josep";
long mida;
```

Variables locals i **variables globals**. S'entén per variable local aquella que es definida dins una funció (setup(), loop(), o qualsevol funció definida per nosaltres). Les variables locals solament són reconegudes des de dins la funció on es troba i desapareixen en acabar-se l'execució de la funció. Però, si necessitem que una variable sigui accessible des de qualsevol lloc del programa i dins de qualsevol funció, caldrà declarar-la com a global. Això és fa declarant la variable fora de qualsevol funció.

Els **operadors** són els que indiquen operacions a efectuar sobre les dades, constants i/o variables. Aquests és classifiquen en: *assignació* (=), *aritmètics* (+, -, *, /, %, etc.), *lògics* (&&, ||, !), *de comparació* (>, <, ==, !=, etc.), *compostos* (++, --,+=, -=, *=, %=, etc.), etc.

4.4.- Estructures bàsiques de programació.

L'execució d'un programa s'efectua executant les instruccions de forma seqüencial, una rere l'altra, seguint l'ordre en que estan escrites, però, en ocasions es convenient aplicar variacions en aquesta seqüència mitjançant el condicionals i els bucles. Les estructures bàsiques per a condicionals i bucles són les següents:

Condicional: if ... else, switch ... case
Bucles: for, while, do...while, break, continue, etc.

if i **if ... else**

Introdueix un condicional, si la condició és certa es fa una determinada acció, sinó, una altra.

```
if(variable > 10) {
     //instruccions a executar en el cas que la condició sigui certa.
}


if(variable > 10) {
     //instruccions a executar en el cas que la condició sigui certa.
}
else {
    //instruccions a executar en el cas que la condició sigui falsa.
}
```

switch .... case

Ens permet fer unes determinades accions segons sigui el valor de una variable. Per a cada cas a considerar, tindrem les accions que li corresponent.

```
switch (variable) {
    case valor1:
      //accions a fer si variable igual a valor1
      break;
    case valor2:
      //accions a fer si variable igual a valor2
      break;
    default:
      // opcions a fer per defecte, si no hi ha coincidència prèvia.
    break;
}
```

for

Repeteix un conjunt de instruccions un determinat nombre de vegades.

```
for (inicialització variable comptadora; condició acabament; in/decrement de la
variable) {

        //instruccions a repetir;

}

Exemple:

for (int i=0; i<100; ++i) {

      //instruccions a repetir;

}
```

while

Repeteix un conjunt de instruccions un nombre de vegades que, inicialment, pot ser indeterminat.

```
while(condicio) {
    //instruccions a repetir
}

int a;
while(a<10) {
    //.....
   ++a;
}
```

do ... while

Repeteix un nombre de instruccions fins o mentre que, una condició sigui certa.

És similar a while, però, ara es garanteix que el contingut del bucle es repetirà, al menys, una vegada.

```
do {
     //instruccions a repetir
} while (condició)
```

break
Executada dins un bucle (for, while, do...while), aquesta instrucció provoca la finalització i sortida del bucle, independentment de les condicions del bucle.

```
for (var a=10; a>0; --a) {
    if(a==3) break; //solament veurem els valors entre 10 i 4.
    Serial.println(a);
}
```

continue
Executada dins un bucle (for, while, do...while), aquesta instrucció finalitza la iteració actual i força l'inici d'una nova iteració.

```
for (var a=10; a>0; --a) {
    if(a%2==0) continue; //solament veurem els valors senars entre 10 i 1
    Serial.println(a);
}
```

4.5.- Funcions.

Les funcions són blocs de codi que executen una determinada acció o operació. Les funcions és refereixen segons el seu nom de definició i poden rebre paràmetres i/o retornar valors al programa invocant de la funció.

Definirem funcions en un programa quan tinguem blocs de codi que s'hagin de repetir des de diferents llocs del programa, o bé, facin operacions específiques que vulguem programar a part del programa principal, com a conseqüència de desitjar un format estructurat en un programa complex.

La sintaxis de la funció és: `tipus nomFunció(tipus param1, tipus param2, etc.)`

> `tipus` és el tipus de dades que retorna la funció, si posem void no retornarà res. També és per indicar el tipus de dades del paràmetre.
> `paramN` és el paràmetre que passem com a dada a la funció, són opcionals i en cas que no hi hagi paràmetres, cal mantenir els parèntesis.

A continuació de la funció escriurem les instruccions que la formen entre claus **{}**
Dins de la funció, finalitzarem amb `return` seguit d'un valor, en cas que la funció hagi de retornar un valor.
Un cop definida la funció, la podrem invocar des de qualsevol altre lloc del programa, passant-li els paràmetres que requereixi i ella retornarà un valor (si de cas) resultant o bé, efectuarà unes accions.

Exemple:
Definició de la funció:

```
int calculaSuma(int val1, int val2) {
        int resultat;
        resultat= val1+val2;
        return resultat;
}
```

Invocació de la funció:

```
Serial.print("La suma de 3 i 2 val: ");
Serial.println(calculaSuma(3, 2));
```

4.6.- Funcions predefinides

El llenguatge de programació C implementat a Arduino incorpora tot un conjunt de funcions predefinides per defecte, que podem utilitzar per als tractament dels diferents elements d'E/S que formen el sistema. Les més generals estan carregades per defecte i, altres més específiques cal carregar-les prèviament mitjançant les llibreries.

Tenim diverses categories de funcions: E/S digital, E/S analògica, Temps, matemàtiques, de tractament de caràcters, etc. Algunes de les més principals es troben descrites a continuació, però, la resta d'elles, més específiques, en trobarem informació a la web de Arduino: https://www.arduino.cc/en/Reference/HomePage.

Funcions predefinides de **E/S digital**:

> **pinMode(pin, mode)** – Utilitzada per inicialitzar el si un determinat pin serà usat com entrada o sortida.
> **digitalWrite(pin, value)** – Assigna un valor lògic a un pin que ha estat definit prèviament com a sortida.
> **digitalRead(pin)** – Llegeix l'estat lògic del pin que ha estat definit prèviament com a entrada i en retorna el valor llegit.

Funcions predefinides de **E/S analògica:**

> **analogRead(pin)** – Llegeix el valor analògic (d'entre 0 i 1023) d'un dels pins d'entrada analògica (A0 ... A5).
> **analogWrite(pin, value)** – Escriu un valor analògic (entre 0 i 255) a un dels pins que suporta sortida PWM i hagi estat inicialitzat com a sortida, per generar un senyal PWM amb un DC corresponent al valor indicat (si value=0 → DC=0%, si value=255 → DC=100%).

Funcions predefinides **de temps:**

> **delay(ms)** – Provoca l'aturada del programa durant un temps, en milisegons, indicat per ms.
> **millis()** - Retorna el nombre de milisegons des de l'inici del programa.

Funcions predefinides **de matemàtiques:**

> **pow(base, exponent)** – Calcula la potència de un numero (base) elevat a (exponent).
> **sqrt(x)** – Calcula l'arrel quadrada del numero indicat a x.
> **map(valor, fromLow, fromHigh, toLow, toHigh)** – re-mapeja el número indicat per *valor* des del rang *fromLow-fromHigh* al rang *toLow-toHigh*.
> **randomSeed(seed)** – inicialitza el generador de números aleatoris. *seed* és la llavor i es pot agafar d'una font aleatòria, per exemple llegir el valor de "soroll" d'una entrada analògica desconnectada.
> **random(min, max)** – retorna un valor numèric aleatori entre *min* i *max*-1.

Funcions predefinides **de tractament d'interrupcions:**

> L'execució seqüencial del codi dins de loop() pot testejar contínuament el valors dels pins d'entrada i aplicar valors a les sortides, però, això requereix un temps de realització de cada cicle de loop() i pot fer que determinades operacions no es puguin atendre immediatament pel retard del bucle. Si és necessari atendre immediatament un esdeveniment determinat podem implementar-lo com a interrupció (pins 2 i 3 de l'Arduino Uno). Per això tenim les funcions que associen el pin amb una funció que atendrà el tractament de la interrupció.

attachInterrupt(digitalPinToInterrupt(pin), ISR, mode) – Vinculem un pin (2 o 3 al Arduino Uno) a una funció ISR (Interrupt Service Routines) que s'encarregarà d'atendre la interrupció quan s'origini de la forma indicada per mode (nivell alt, flanc ascendent o flanc descendent).
detachInterrupt(digitalPinToInterrupt(pin)) – Desvinculem un pin de l'atenció d'interrupcions definida prèviament amb attachInterrupt().
interrupts() - Habilita l'atenció a interrupcions que hagin estat associades amb attachInterrupt().
noInterrupts() - Inhabilita l'atenció a interrupcions que hagin estat associades amb attachInterrupt().

Funcions predefinides **de comunicació sèrie:**
Serial.begin(speed) – Inicialitza la comunicació en sèrie amb el monitor sèrie disponible en el IDE speed és la velocitat en bps, generalment 9600bps.
Serial.available() - Retorna el nombre de bytes disponibles per agafar des del buffer del port sèrie, si no n'hi ha, retorna 0.
Serial.println(val) – Escriu el valor val al monitor sèrie més un salt de línia.
***Serial*.readString()** - Llegeix i retorna el text llegit des del monitor sèrie.

Funcions predefinides **de tractament de cadenes (String):**
Aquestes funcions s'apliquen en el tractament de dades tipus alfanumèriques contingudes en estructures de dades tipus String. També existeixen funcions pel tractament de cadenes contingudes en estructures de dades string formades en arrays de char's. Alguna mostra per a les dades tipus String són:
concat(cad) - Concatena dos cadenes. És equivalent a + i +=.
indexOf(val) - Cerca la posició d'inici d'una subcadena val dins una cadena.
charAt(n) – Retorna el caracter que es troba en la posició *n* dins la cadena.
substring(inici, fi) – Retorna el tros de cadena entre la posició *inici* i la posició *fi* d'una cadena.
toInt() - Retorna el valor numèric tipus Int d'una cadena, si conté dígits numèrics.

Existeixen també moltes altres funcions predefinides que no estan disponibles per defecte. Són les que es carreguen amb la inclusió d'una llibreria específica. Per conèixer quines són en cada cas, cal trobar la informació detallada de la llibreria carregada.

4.7.- Mètodes Polling i Interrupt.

Aquests mètodes són propis de la programació en sistemes microprocessadors i microcontroladors. El programa dels circuits electrònics programats tenen com a funció habitual la lectura continuada dels sensors que introdueixen informació al programa. Aquesta lectura s'efectua de forma cíclica dins de la funció *loop()*, cada cicle s'anomena scan, i, en funció dels valors llegits, es deriva a la realització d'unes tasques o altres per atendre als valors d'entrada al sistema. Aquest mode consistent en la consulta continua i seqüencial dels sensors es coneix amb el nom de **Polling**.

Cal tenir en compte que el temps entre lectures per a cada sensor, d'un scan al següent, és variable i imprecís, ja que depèn de les tasques que hagi de fer el microcontrolador segons els valors llegits. A més, si el programa utilitza funcions com *delay()* i similars, aquest temps encara pot ser més gran i variable. Tot i això, el mètode *Polling* és vàlid per la majoria d'aplicacions.

En ocasions, però, podem tenir que determinats esdeveniments és produeixen durant un temps molt curt i de forma aleatòria. Si en el moment en que passa l'esdeveniment el microcontrolador està ocupat fent una altra tasca (per exemple, esperant temps per un *delay()*, esperant la recepció d'una dada, esperant la confirmació d'una acció feta a un dispositiu perifèric de sortida, etc.) es perdrà l'ocasió d'atendre la informació que aporta l'esdeveniment produït i ja no serà recuperable.

Com a exemple d'esdeveniments ràpids podem tenir: detectar la producció d'un senyal fort d'àudio que dura uns pocs µs o ms, o bé, detectar un canvi d'estat puntual d'un nivell lògic, o bé, detectar el pas per zero d'una ona sinusoïdal, etc.

En tots aquests casos d'exemple és més apropiat utilitzar el mètode *Interrupt*, enlloc del *Polling*, si es que no volem perdre informació com a conseqüència del temps d'scan.
El mètode **Interrupt** consisteix en que, en produir-se l'esdeveniment, es genera una interrupció que atura el programa (sempre que tingui més prioritat) que s'està executant dins de loop(). El programa s'atura en un estat conegut i es passa el control a un altre bloc de codi que s'encarrega expressament d'atendre el sensor que origina la interrupció, invocant una funció específica. En acabar d'atendre la interrupció, es retorna a l'execució del codi que s'estava executant abans de l'aparició de la interrupció.

Les interrupcions poden ser originades per dispositius interns al microcontrolador (timers, watchdogs, etc.) o per dispositius externs. En ocasions, segons quin dispositiu sigui que l'origina, les interrupcions és poden produir amb diferents nivells de prioritat.
En el cas de Arduino Uno, les interrupcions externes són detectades als pins 2 i 3.

5.- Pràctiques.

En aquest apartat es descriu un conjunt de pràctiques resoltes en les que es mostra exemples d'utilització i aplicació d'Arduino.

A cada pràctica s'introdueix algun concepte nou, diferent dels anteriors, intentant que sigui el més aïllat possible per no mesclar-los i fer-los més entenedors, però, cadascuna de les pràctiques pot ser modificada, resolent el mateix de forma més òptima, o, ser ampliada amb noves funcionalitats.

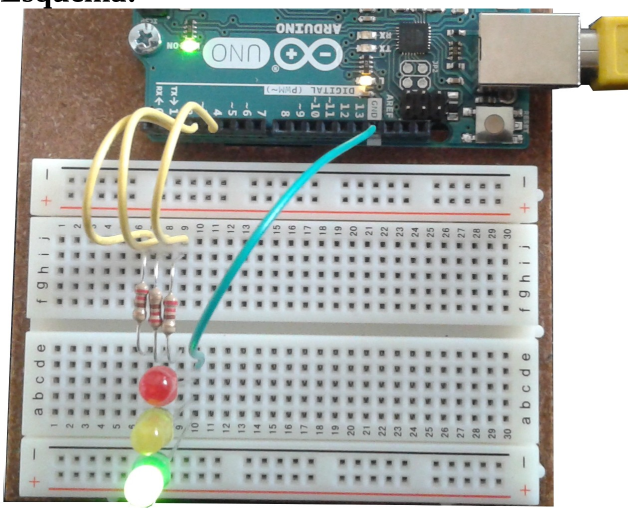
Amb imaginació és pot arribar a implementar moltes variants sobre la mateixa pràctica i obtenir-ne resultats més engrescadors. Depèn de cadascú, segons les seves possibilitats, millorar i expandir les diferents pràctiques per assolir el propi objectiu desitjat.

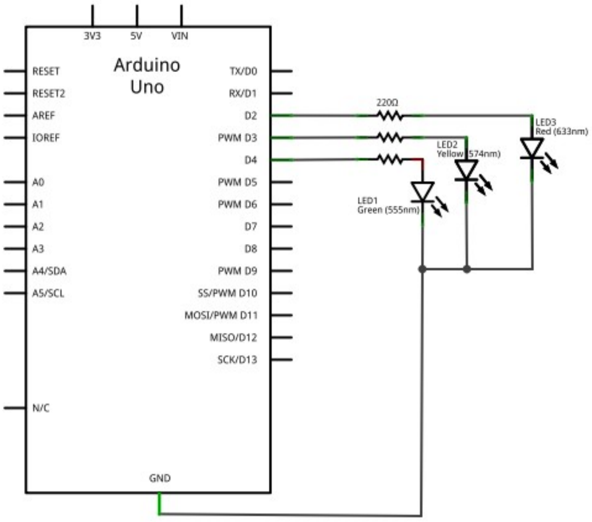
5.1.- Pràctica 1. Semàfor.

Crear un semàfor que il·lumini tres leds de colors verd, groc i roig. La seqüència ha de començar pel verd, després el groc i finalment el vermell, però, poc temps abans del canvi de groc a vermell, el groc ha de parpellejar. La seqüència s'ha de repetir cíclicament.

Components: 3 leds roig, groc i verd, 3 resistències de 220 Ω.

Esquema:

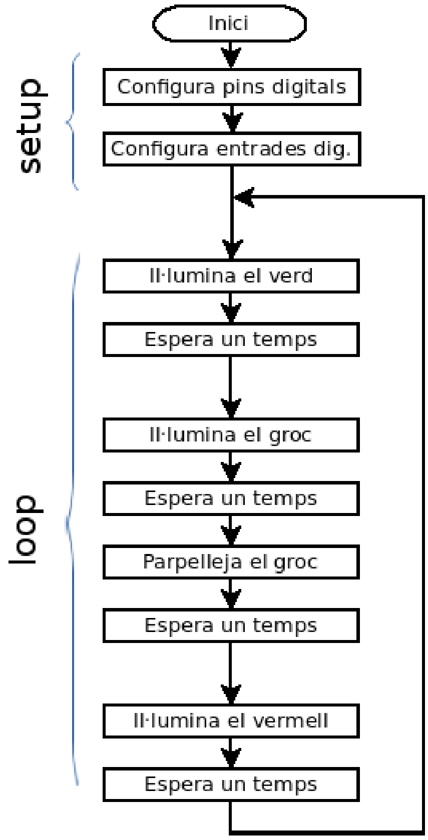




Programa:

```
/* Semàfor. Led Roig a sortida 2, groc a 3
i verd a 4.
 *
 */
int temps;
void setup() {
  // inicialitza sortides digitals.
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  temps=1000;
}

void loop() {
  //il·luminem solament el verd:
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  delay(temps*4);
  //il·luminem el groc:
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  delay(temps*3);
  //parpelleig de groc:
  for(int i=0; i<10; ++i) {
    if(i%2) digitalWrite(3, HIGH);
    else  digitalWrite(3, LOW);
    delay(temps*0.2);
  }
  //il·luminem el vermell:
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  delay(temps*4);
}
```

Algorisme:



5.2.- Pràctica 2. Semàfor controlat d'un pas de vianants.

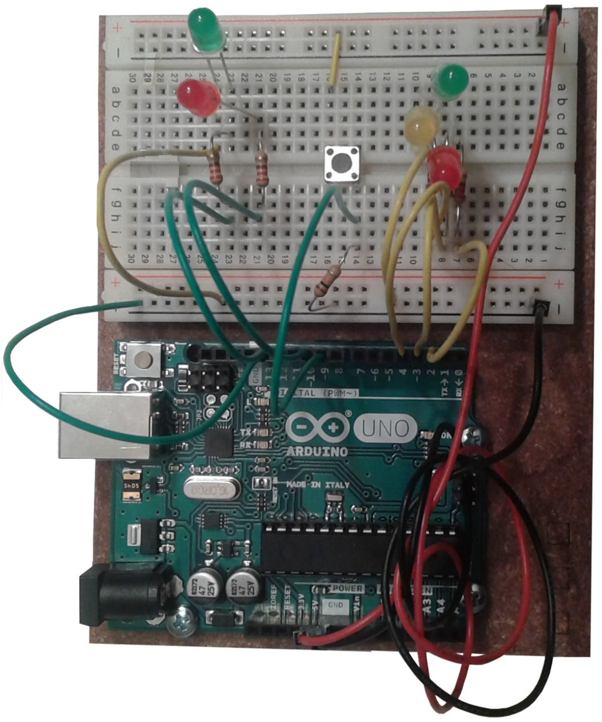
Tenim un pas de vianant on hi ha dos semàfors, un per vehicles (roig, groc i verd) i un per vianants (roig i verd). En condicions normals els dos semàfors efectuen la seqüència normal de facilitar el pas per vehicles i aturar els vianants i a la inversa de forma cíclica, en el canvi de groc a vermell pels vehicles o de verd a vermell per als vianants, els respectius colors parpellegen per indicar el canvi proper.

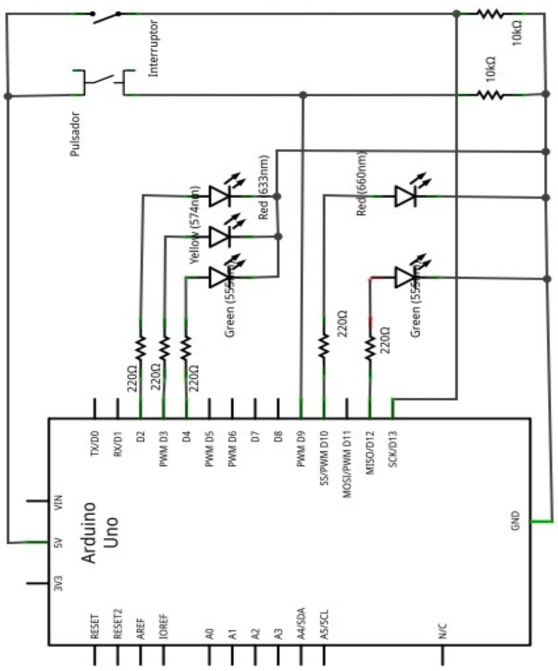
Com que en el pas de vianants hi ha poca afluència de vianants, es disposa d'un interruptor (manipulable solament pels agents de control de trànsit) que, quan està activat (a 1) fa que el semàfor de vehicles sempre estigui a verd i el de vianants sempre a vermell, per tant, mai s'atura als vehicles i no es permet passar als vianants. Però, quan està activat aquest mode on no s'atura als vehicles, també es disposa d'un polsador per a vianants que, en preme'l, inicia, al cap d'un temps curt aleatori, i per una sola vegada, la seqüència d'aturada de vehicles i pas per als vianants.

Crear, mitjançant leds de color vermell, verd i groc, polsador, interruptor i altres components auxiliars, el circuit i el programa que implementi el funcionament indicat.

Components: 2 leds roig, 1 de groc i 2 de verd, 5 resistències de 220 Ω. Un polsador, un interruptor i dos resistències de 10kΩ.

Esquema:





Programa:

```
/* Semàfor vehicles. Led Roig a sortida 2,
groc a 3 i verd a 4.
 * Semàfor vianants. Led Roig a sortida
10, verd a 12.
 * Polsador vianant. Entrada:9.
Interruptor mode control Entrada 13.
 *
 */
int temps;
void setup() {

  // inicialització pins digitals.
  pinMode(2, OUTPUT); //vehicles roig.
  pinMode(3, OUTPUT); //vehicles groc.
  pinMode(4, OUTPUT); //vehicles verd
  pinMode(10, OUTPUT); //vianants roig
  pinMode(12, OUTPUT); //vianants verd
  pinMode(9, INPUT);  //entrada polsador
vianants
  pinMode(13, INPUT); //entrada
interruptor de mode
  temps=1000;
}

void loop() {

  int intAgents=digitalRead(13);
//comprovo el mode de treball, si 0 mode
automatic, si 1 mode polsador.
  int pulsVianant=0;
  if(intAgents==1)
pulsVianant=digitalRead(9); //en cas mode
1 polsador, comprovo polsador de vianant
  digitalWrite(4, 1); //vehicles verd.
Mode 0.
  digitalWrite(2, 0);
  digitalWrite(10, 1); //vianants roig
  digitalWrite(12, 0);
  if(intAgents==0||pulsVianant==1) { //al
mode 0 es fa la sequencia automàtica i si
es prém també.
      delay(temps*4);
      digitalWrite(4, 0); //vehicles verd
      digitalWrite(3, 1); //vehicles groc
      delay(temps*2);
      for(int i=0; i<10; ++i) {
       if(i%2==0) digitalWrite(3, 0);
//vehicles groc blink
       else digitalWrite(3, 1);
       delay(temps*0.2);
      }
      digitalWrite(3, 0);
      digitalWrite(2, 1); //vehicles roig
      digitalWrite(10, 0);
      digitalWrite(12,1); //vianants verd
      delay(temps*4);
      for(int i=0; i<10; ++i) {
       if(i%2==0) digitalWrite(12, 0);
```

Declarem la variable temps fora de qualsevol funció per que sigui un variable global, per tant accessible des de dins de les funcions *setup()* i *loop()*.

Dins de *setup()* inicialitzem les sortides digitals i el valor de la variable temps.

Primer determinem el mode de treball (automatic 0 o polsador 1) i ho a pulsVianant.

La seqüència automàtica consisteix en l'alternança entre els semàfors de vehicles i de vianants.

```
//vianants verd blink
        else digitalWrite(12, 1);
        delay(temps*0.2);
       }
   }
}
```

S'observa en el funcionament del programa que la detecció del polsador no és instantània i cal mantenir-lo premut durant una estona per que el canvi d'estat sigui efectiu. Això és per que utilitzem la instrucció "delay()" dins el bucle "loop()" que atura el bucle i retarda l'instant de detecció de la polsació. En les pràctiques següents es veu una altra manera d'introduir retards per a alguns blocs dins el bucle, però, sense que s'hagi d'aturar el "loop()" (amb millis() o interrupt()) i no afectar la detecció d'altres elements dins el bucle, al contrari del que passa en aquest programa.

5.3.- Pràctica 3. Termòstat electrònic.

Un termòstat és un dispositiu que detecta la temperatura ambient del seu entorn i la compara amb la temperatura prefixada per l'usuari com a consigna. En funció de la comparació de temperatures efectuada, s'activarà, o no, el sistema de calefacció.

Els termòstats convencionals consten d'un sistema mecànic format per un bimetall que, en variar la temperatura, aquest es deforma per efecte de la dilatació i acciona, o no, segons correspongui a la temperatura de consigna indicada per l'usuari, un interruptor de connexió o desconnexió del sistema de calefacció.

En aquesta pràctica implementarem un termòstat electrònic i programat amb el mateix funcionament descrit prèviament.
Detectarem la temperatura ambient mitjançant el sensor de temperatura TMP36 i prefixarem la temperatura desitjada mitjançant un potenciòmetre entre el rang de 10ºC i 30ºC. Si la temperatura ambient es inferior a la temperatura desitjada, activarem el sistema de calefacció, de cas contrari, l'aturarem. Un cop s'ha activat el sistema de calefacció, aquest haurà de funcionar un mínim de 5', encara que la temperatura en sigui superior, per evitar les activacions i desactivacions continues en límit de la comparació. L'activació, o no, del sistema de calefacció el simularem mitjançant un Led.

Components: 1 Led, 1 R de 220Ω, 1 potenciòmetre de 10KΩ i un sensor de temperatura TMP36.

El sensor de temperatura TMP36 ens subministra una tensió proporcional a la temperatura de 10mV/ºC. Com que s'alimenta el sensor TMP36 amb una tensió de 5v, el rang de tensions que ens pot subministrar estarà entre 0 i 5v (teòricament, ja que a partir d'uns determinats límits de temperatura ja no serà efectiu). El valor d'aquesta tensió es convertira a digital mitjançant el convertidor A/D incorporat en les entrades analògiques, concretament, utilitzarem l'entrada A0. El CA/D és de 10 bits, per tant disposarem de 2^{10}=1024 valors digitals possibles (entre 0 i 1023).

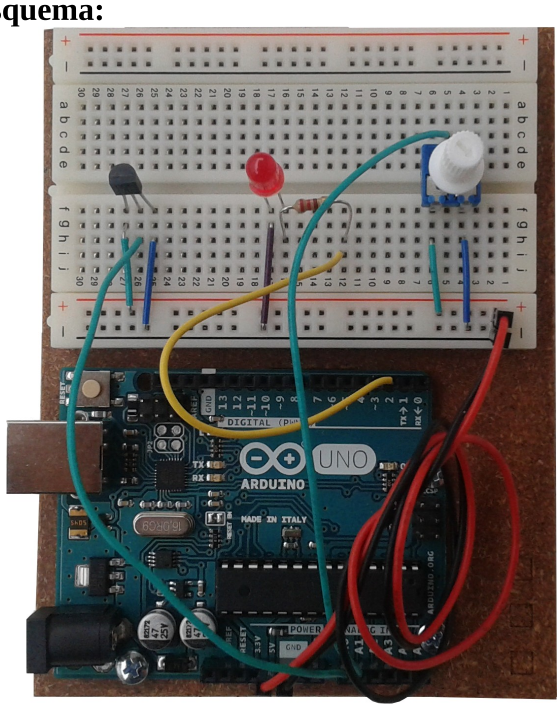
Per obtenir el valor numèric de la tensió introduïda a A0 apliquem la fórmula: Vimax*A0/n, on Vimax és la màxima tensió possible a obtenir del sensor (5v en el nostre cas), A0 representa el valor digital convertit i n és el màxim nombre de valors que pot distingir el CA/D (1024 en el nostre cas).
Un cop tenim el valor de la tensió llegida (del datasheet: 25ºC → 750 mV), obtindrem el valor de la temperatura en ºC amb la fórmula: (v-0'5)*100.

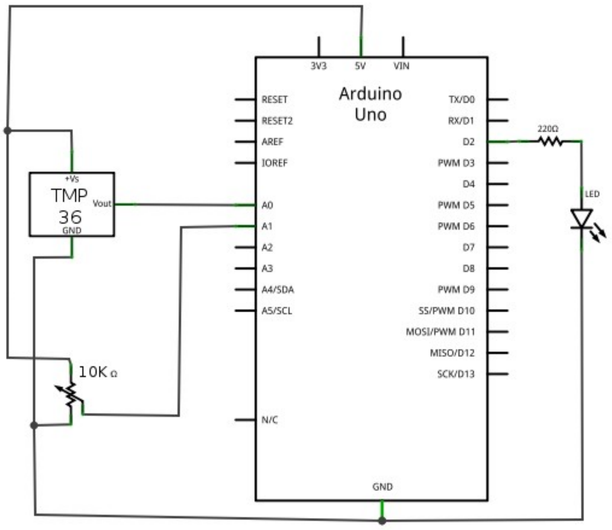
Respecte a la introducció de la temperatura de consigna, utilitzem un potenciòmetre amb 5 v aplicats als seus extrems, del cursor n'obtindrem una tensió proporcional entre 0 i 5v. Mitjançant l'aplicació d'aquesta tensió al CA/D de l'entrada A1 obtindrem un valor numèric proporcional a aquesta tensió en un rang de 0 a 1023. Per obtenir el valor equivalent a la temperatura de consigna indicada entre els marges de 10ºC a 30ºC apliquem la fórmula: ((30-10)/1024)*A0+10, on el quocient és la pendent d'una recta i +10 representa el offset respecte de l'origen. A0 és el valor digital llegit a l'entrada analògica A1.

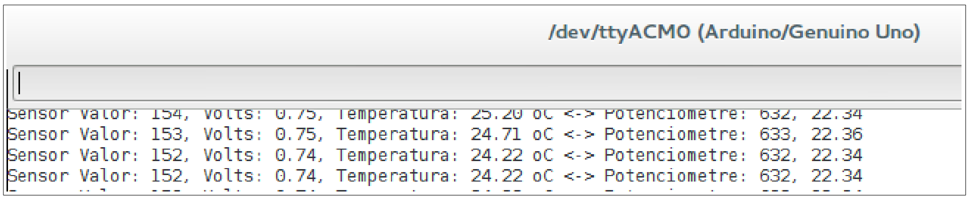
Si obrim la consola Serie en el IDE, podrem veure els valors mesurats i calculats en el sensor i el potenciòmetre, de forma contínua i en temps real.

El valor mesurat pel sensor i l'indicat amb el potenciòmetre els comparem per determinar si cal engegar la calefacció, o no, simulada mitjançant un llum Led.

Esquema:







Programa:

```
void setup() {
  pinMode(2, OUTPUT);
  Serial.begin(9600);
}
```

Dins de la funció setup() inicialitzem el mode del pin 2 i la comunicació sèrie.

```
void loop() {
  //Llegim valor sensor temperatura:
  int sensTMP36=analogRead(A0); //Valor
binari del CA/D (0-1023)
  float v=(sensTMP36/1024.0)*5.0; //Obtenim
el voltatge equivalent
  float t=(v-0.5)*100; //Obtenim la
Temperatura equivalent;
  //Llegim el potenciometre:
  int pot=analogRead(A1); //Valor binari
CA/D (0-1023)
  //conversio al rang entre 10°C-30°C:
  float tpot=((30.0-10.0)/1024.0)*pot+10.0;

  String text="Sensor Valor: ";
  text.concat(sensTMP36);
  text.concat(", Volts: ");
  text+=v;
  text+=", Temperatura: ";
  text+=t;
  text+=" oC <-> Potenciometre: ";
  text+=pot;
  text+=", ";
  text.concat(tpot);
  Serial.println(text);
  delay(1000);

  if(t<tpot) {
    digitalWrite(2,1);
    delay(5000);
  }
  else digitalWrite(2,0);
}
```

Les variables definides com a tipus float contindran valors numèrics amb decimals.

La variable text acumularà un text concatenat que s'enviarà per la consola sèrie en una sola línia.

En el cas que efectuem l'activació de la calefacció, esperarem un temps de 5" abans de repetir el cicle, de cas contrari, no.

5.4.- Pràctica 4. Sensor de llum.

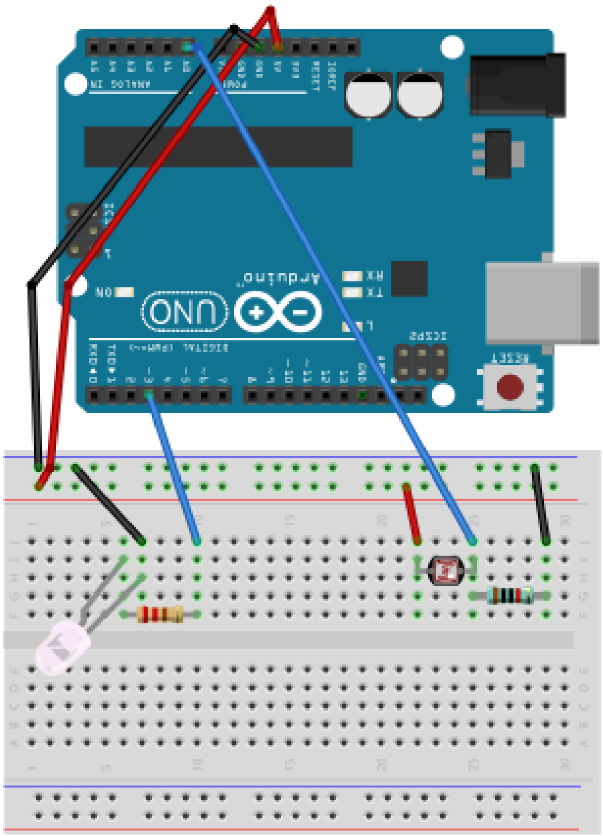
Volem utilitzar un resistor LDR per detectar el nivell de llum. Si aquest nivell és inferior a un determinat valor, encendrem un llum LED, de cas contrari, l'apagarem. Aquest és un circuit típic sensor de llum que pot ser utilitzat com a interruptor crepuscular (encendre el llum automàticament quan es fa fosc).

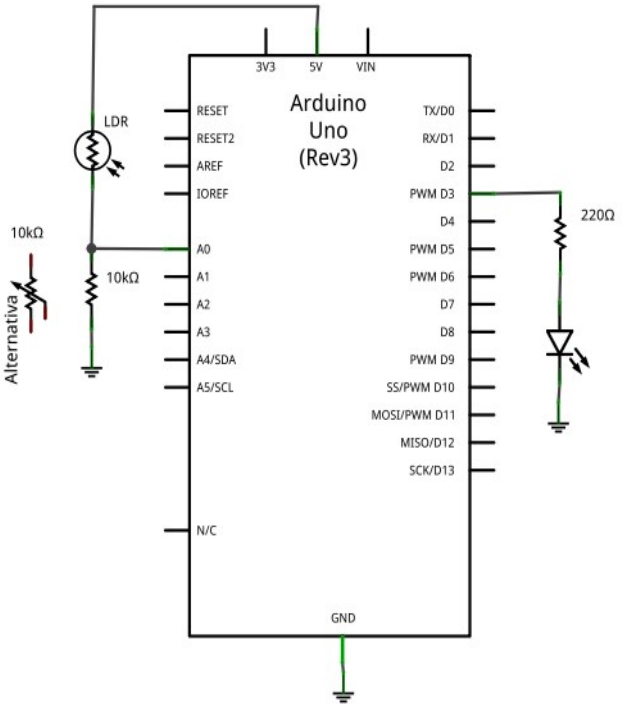
El resistor LDR disminueix la seva resistència quan més llum incideixi sobre ella. En l'esquema podem veure la LDR en sèrie amb la resistència de 10kΩ, aquestes dos formen un divisor de tensió. La tensió del divisor de tensió s'aplica a l'entrada analògica A0, i en ella, per programa llegim un valor numèric del C A/D entre 0 i 1023, corresponents al voltatge entre 0 i 5v.
Llavors podem determinar el valor digital corresponent al nivell de llum que volem que faci canviar el valor d'il·luminació del LED.

A l'esquema es pot veure el símbol alternatiu d'un resistor ajustable de 10kΩ que pot substituir al resistor fix de 10kΩ. Aquestes són dos alternatives: l'una ens permet ajustar manualment el valor del llindar de canvi del llum i l'altra no (evidentment, sempre podrem canviar-ho per programa).

Components: 1 fotoresistor, 1 Led, 1 resistor de 10kΩ, 1 resistor de 220Ω, alternativament, també un resistor variable de 10kΩ.

Esquema:





El programa defineix unes constants LED i LDR que seran usades posteriorment. En la funció *setup()* s'inicialitza el pin i la comunicació de la consola sèrie. La consola sèrie s'utilitzarà com a auxiliar per poder llegir, de forma contínua el valor del sensor LDR i poder determinar quin valor cal utilitzar com a nivell del llindar de canvi de l'estat del Led.

En el programa es pot veure múltiples blocs de codi que estan "comentats" (entre /* i */). Això és per provar múltiples formes de funcionament del sensor de llum, però, solament un d'ells pot estar activat, per tant, cal "descomentar", treure /* i */ del bloc que es vulgui provar i mantenir els altres "comentats" per evitar interferències.

El valor llegit del sensor el desem a la variable *sensLdr* que serà utilitzat posteriorment en els diferents blocs de prova. Aquest valor també el podrem veure per la consola sèrie i ens ajudarà a determinar els valors de commutació.

Els diferents blocs a provar són els següents:

- Control digital tot o res: El led s'il·lumina quan la llum ambient es troba sota un determinat nivell i s'apaga quan està per sobre. En aquest cas, pot passar que quan la llum ambient estigui al voltant del llindar de canvi, el led parpellegi contínuament.

- Control digital tot o res, amb histèresi[3]: Aquesta opció evita el parpelleig del led indicat en el cas anterior, quan el valor es proper al llindar, afegim uns marges per sobre i sota del nivell de canvi i evitem la incertesa produïda en el moment de canvi.

- Control analògic inversament proporcional per trams: En aquest cas, farem que el Led s'il·lumini mitjançant un valor analògic obtingut del senyal PWM, amb un valor corresponent a cada tram de valor de llum ambiental.

- Control analògic inversament proporcional: En aquest cas, aplicarem al Led una senyal PWM que farà que s'il·lumini de forma inversament proporcional a la llum ambient. Per determinar el DC del senyal PWM ho fem mitjançant la inversió de la conversió de l'entrada analògica de 1024 valors al rang de 256 valors de la sortida analògica (PWM).

Programa:

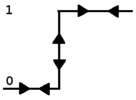
```
#define LED 3
#define LDR A0

void setup() {
  pinMode(LED,OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int sensLdr=analogRead(LDR);
  Serial.println(sensLdr);

  //Control digital Tot o res:
  /*if(sensLdr<200) digitalWrite(LED, HIGH);
  else digitalWrite(LED, LOW);*/

  //Control digital Tot o res amb histèresi:
  /*if(sensLdr<200) digitalWrite(LED, HIGH);
  if(sensLdr>300) digitalWrite(LED, LOW);*/

  //Control analògic inversament proporcional per trams:
  /*if(sensLdr<200) analogWrite(LED, 255);
  else if(sensLdr>=200 && sensLdr<250) analogWrite(LED,
200);
  else if(sensLdr>=250 && sensLdr<350) analogWrite(LED,
150);
  else if(sensLdr>=350 && sensLdr<450) analogWrite(LED,
100);
  else if(sensLdr>=450 && sensLdr<550) analogWrite(LED,
75);
  else if(sensLdr>=550 && sensLdr<650) analogWrite(LED,
25);
  else analogWrite(LED, 0);*/

  //Control analògic inversament proporcional:
  /*int val=int(sensLdr*255/1024);
```

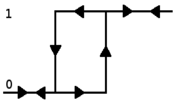
Definim valors per referencia el pin del Led i de la Ldr.

Dins de *setup()* inicialitzem el pin Led i la consola sèrie.

Dins de *loop()* llegim el valor de la Ldr i el mostrem per la consola sèrie.

Els blocs següents cal descomentar-los un a un per provar el funcionament i l'efecte de cadascun d'ells, eliminant els /* */ del bloc a provar.

3 Canvi de nivell sense histèresi: 

, canvi de nivell amb histèresi: 

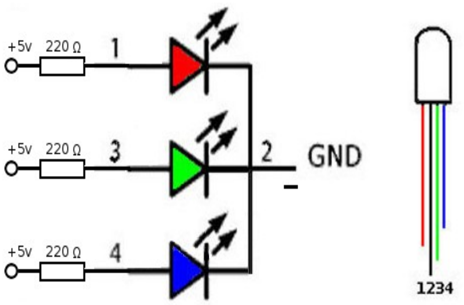
. En el primer cas poden haver imprecisions instantànies, ja que el canvi de connexió i desconnexió es fa amb el mateix llindar. En el segon cas tenim dos llindars, un de connexió i un altre de desconnexió que evita les imprecisions.

```
  val=255-val;
  analogWrite(LED,val);
  //també podem fer el mapejat amb:
map(val,0,1023,0,255);*/

}
```

5.5.- Pràctica 5. Llum multicolor. So multi to.

En aquesta pràctica veurem la utilització del Led RGB per obtenir diferents colors i el brunzidor piezoelèctric per generar tons acústics. Amb el programa implementat, es reproduirà, a l'inici, una melodia, per una sola vegada, i posteriorment s'iniciarà una seqüència contínua d'il·luminació del Led RGB amb diferents colors predefinits.

Components: 1 Led RGB de càtode comú, 3 resistències de 220, 1 brunzidor piezoelèctric.



El **Led RGB** està format per tres leds independents de colors roig, verd i blau encapsulats conjuntament. La mescla additiva de les tres llums dels colors indicats, en diferents proporcions, pot originar un espectre de colors molt ampli.
La imatge lateral ens mostra l'esquema d'un Led RGB de càtode comú i el seu connexionat.

El **transductor piezoelèctric** és un dispositiu de caracter capacitiu i fonamenta el seu funcionament en la piezoelectricitat. L'efecte piezoelèctric consisteix en la capacitat que tenen certs cristalls piezoelèctrics per deformar-se mecànicament quan se'ls aplica una tensió, també a la inversa, si se'ls aplica una pressió mecànica generen una tensió. La deformació mecànica i la tensió són valors molt petits, però, suficients per utilitzar-lo com a brunzidor en aquesta pràctica. Si al cristall piezoelèctric li apliquem una tensió variable, la deformació mecànica del cristall també serà variable i originarà un so corresponent a la freqüència de la variació de la tensió. Per accionar el so en el brunzidor tenim les funcions:

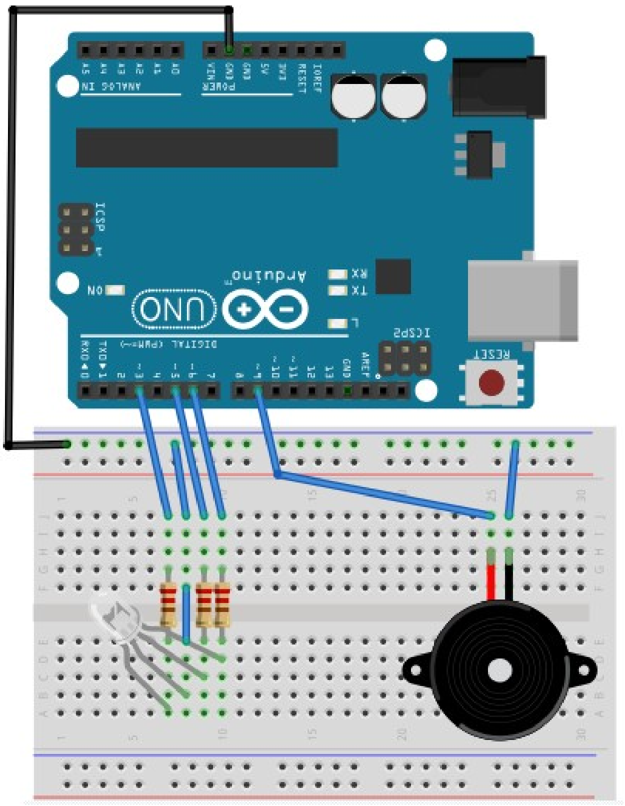
`tone(pin, freq)` – el brunzidor connectat a *pin* emet un so a la freqüència indicada per *freq*.

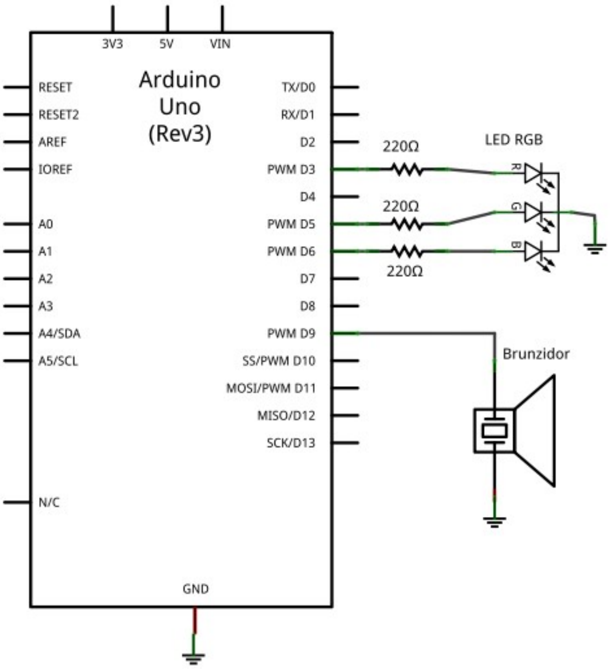
`noTone(pin)` – el brunzidor connectat a *pin* atura el so.

La freqüència del so que s'emet és el d'una ona rectangular, ja que la sortida emprada és digital.

La reversibilitat de l'efecte piezoelèctric fa que aquest dispositiu també pugui ser usat com a micròfon, sensor de so, ja que les ones de so (que són vibracions mecàniques de l'aire) pressionen sobre el cristall i aquest reacciona generant una tensió analògica proporcional al so. S'anomena micròfon piezoelèctric.

Esquema:





Programa:

```
const int R=3;
const int G=5;
const int B=6;
const int Z=9;

int escalaTons[] = {261, 277, 294, 311, 330,
349, 370, 392, 415, 440};
//cançó: Array sequencia de index de
l'escalaTons, acabats amb -100, els negatius
indiquen temps de silenci. Descomentar 1 sola
melodia:
int canso[] = {0, 5, 7, 9, 7, 4, -10, 0, 5, 7,
9, 7, -10, -10, 0, 5, 7, 9, 7, 4, -10, 4, 5, 4,
0, -1, 0, -100};
//int canso[]={1,-1,0,-1,0,2,3,3,-10,4,3,-100};

void setup() {
  pinMode(R, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(Z, OUTPUT);

  for(int i=0; i<10; ++i) {
    tone(Z, escalaTons[i]);
    delay(500);
  }
  noTone(Z);
  delay(2000);
```

Inicialment definim unes constants i variables globals, per indicar els pins on s'ha connectat el Led RGB i el brunzidor.

L'array "escalaTons" conté una llista de freqüències de les notes més habituals. Aquestes notes són referenciades en els arrays de les melodies "canso".

Dins de la funció inicial "setup()" inicialitzem els pins com a sortida i reproduïm els tons indicats a l'array "escalaTons".

Finalment, dins aquesta funció s'efectua la reproducció d'una melodia predefinida a l' Array "canso".
La melodia solament es reprodueix una vegada al principi de la connexió elèctrica.

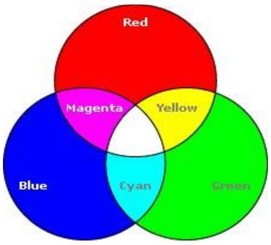
Un cop finalitzada la reproducció de la melodia, s'inicia la representació de colors

```
  int i=0;
  do {
    if(canso[i]>=0) tone(Z,scalaTons[canso[i]]);
    else noTone(Z);
    if(canso[i]>=0) delay(500);
    else if(canso[i]==-10) delay(500);
    else if(canso[i]==-1) delay(25);
    ++i;
  } while (canso[i]!=-100);
  noTone(Z);
}


void loop() {
  for(int i=1; i<=8; ++i) {
    posaColor(i);
    delay(3500);
  }
  tone(Z,12500);
  delay(30);
  noTone(Z);
}


void posaColor(int color) {
  switch(color) {
    case 1: //roig
        analogWrite(R,255);
        analogWrite(G,0);
        analogWrite(B,0);
        break;
    case 2: //verd
        analogWrite(R,0);
        analogWrite(G,255);
        analogWrite(B,0);
        break;
    case 3: //blau
        analogWrite(R,0);
        analogWrite(G,0);
        analogWrite(B,255);
        break;
    case 4: //groc
        analogWrite(R,255);
        analogWrite(G,255);
        analogWrite(B,0);
        break;
     case 5: //cian
        analogWrite(R,0);
        analogWrite(G,255);
        analogWrite(B,255);
        break;
    case 6: //magenta
        analogWrite(R,255);
        analogWrite(G,0);
        analogWrite(B,255);
        break;
    case 7: //blanc
        analogWrite(R,255);
        analogWrite(G,255);
        analogWrite(B,255);
        break;
    case 8: //marro
        analogWrite(R,235);
```

en el Led RGB. Els colors a representar estan definits a la funció "posaColor()".

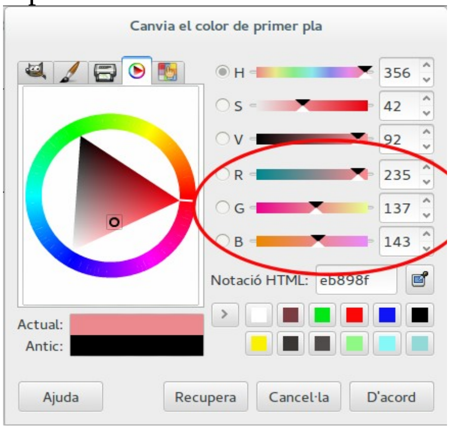
A cada iteració de "loop()" és fa un beep informatiu.
La funció "posaColor()" és la que s'encarrega d'il·luminar el Led per a cada color indicat com a paràmetre de la funció.

La representació d'un color es fa posant un valor analògic entre 0 i 255 per a cadascun dels tres colors primaris del Led. Segons sigui el valor posat, s'obtindrà un color resultant o un altre.



Estan definits 8 colors, però, si es vol posar més colors només cal repetir la mateixa seqüència indicant el valor analògic a posar-hi. Una forma d'obtenir el valor analògic entre 0 i 255 de cada color primari per obtenir el color que volem és amb l'ajuda d'un programa d'edició gràfica, on, en el seu selector de colors ens indica, per a cada color triat, quin valor correspon per a cada color primari.

La següent imatge correspon al cas de Gimp, utilitzada per determinar els valors corresponents al color marró:



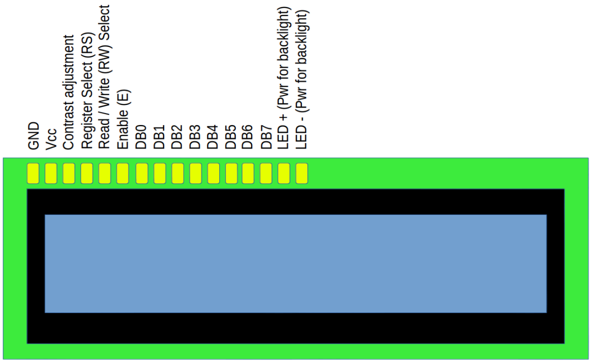
```
        analogWrite(G,137);
        analogWrite(B,143);
        break;
  }
}
```

5.6.- Pràctica 6. Rellotge digital i alarma amb visualitzador LCD.

Mitjançant el visualitzador LCD mostrarem l'hora d'un rellotge. També podrem programar una hora de alarma que ens activarà un brunzidor emetent dos tons. Ajustarem l'hora i l'alarma amb comandes des de la consola sèrie. Mitjançant un polsador activarem, o no, l'alarma.

El **visualitzador LCD** és un dispositiu compost que incorpora una pantalla amb matrius de punts (7x5) per formar caracters alfanumèrics, més un circuit electrònic que implementa el control del LCD i la interfície de comunicació amb altres circuits electrònics. N'hi ha de diversos tipus i mides, d'una o més files i múltiples caracters per fila.

LCD LCM1602C 2x16



Connexionat del LCD LCM1602C, compatible amb: Hitachi HD44780

Per controlar aquest dispositiu LCD des d'un programa Arduino cal carregar la llibreria corresponent:

```
#include <LiquidCrystal.h>
```

Un cop carregada la llibreria, disposem de noves funcions específiques per aquest dispositiu. Amb la sintaxis de la funció:

```
LiquidCrystal  nomLcd(RS, Enable, db4, db5, db6, db7);
```

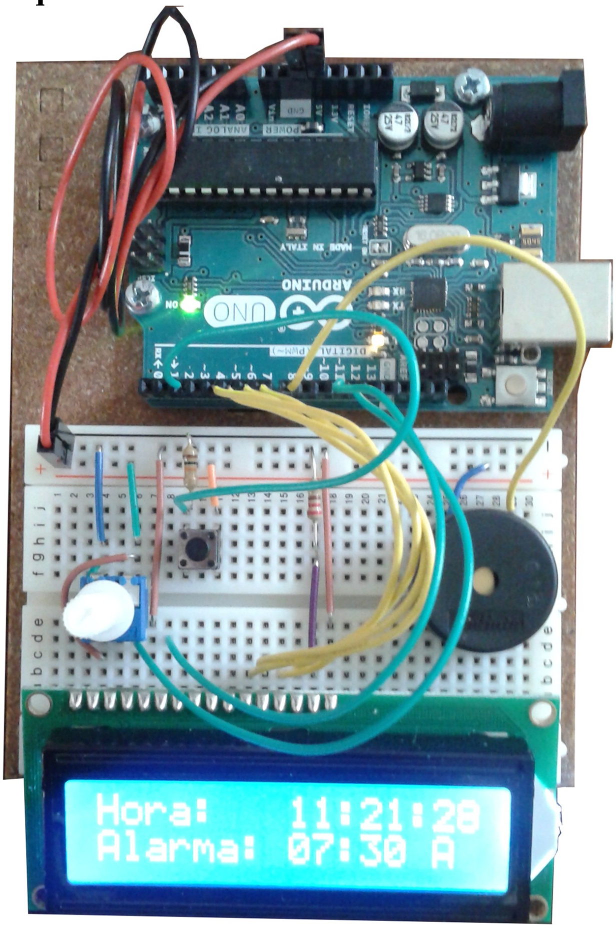
inicialitzem el dispositiu, amb la seva expressió més simple[4], indicant en quines sortides digitals de Arduino han estat connectats els principals pins del display. Un cop inicialitzat es poden utilitzar múltiples funcions específiques afegides per aquest dispositiu, per exemple:

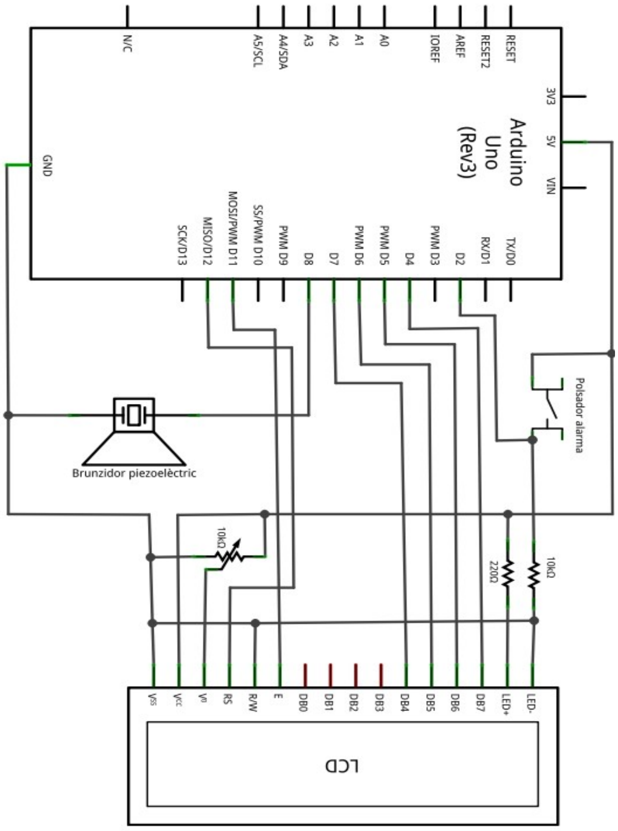
`nomLcd.begin(cols, fils)` indiquem el nombre de columnes i files.
`nomLcd.clear()` esborrem tot el display
`nomLcd.setCursor(col, fil)` posiciona el cursor en una columna, fila.
`nomLcd.print(text)` escriu *text* a partir de la posició del cursor.
etc.

4 Existeix altres formes de inicialització on s'indiquen la connexió de més pins del dispositiu LCD.

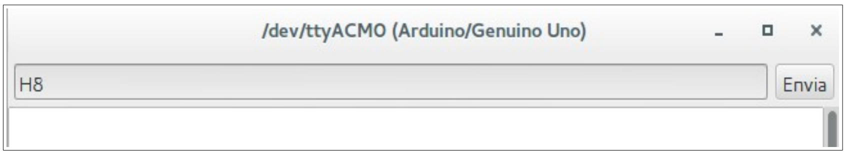
Components: Display LCD LCM1602C (compatible amb: Hitachi HD44780), 1 potenciòmetre de 10kΩ, 1 brunzidor piezoelèctric, 1 polsador, 1 resistència de 10kΩ, 1 resistència de 220Ω, 1 condensador de 100µF.

Esquema:





Per la consola sèrie és on enviarem els missatges al programa per ajustar l'hora i l'alarma. Per ajustar l'hora, els minuts, segons i hora d'alarma i minuts d'alarma ho faríem posant prèviament les lletres: H, M, S, h i m, respectivament, abans del número. A la imatge següent estem posant l'hora (H) del rellotge a les 8.



Programa:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,7,6,5,4);

int h, m, s, ha, ma, activAl, estatAl;
void setup() {
  lcd.begin(16,2); //ncolumnes, nfiles
  pinMode(2,INPUT); //pulsador Activa alarma
  pinMode(8,OUTPUT); //brunzidor piezoel.
```

Inicialment, el programa carrega la llibreria *LiquidCrystal.h* que proveeix del conjunt de funcions específiques d'aquest dispositiu i, posteriorment, crea un objecte

```
  h=m=s=ha=ma=0;
  activAl=false; //indica si l'alarma es operativa
  estatAl=0; //indica el to del so que emet l'alarma
  Serial.begin(9600);
}

void loop() {
  /*Si de cas, llegim de la consola sèrie per posar en
hora
    Hnn-hora, Mnn-minuts, Snn-segons, hnn-hora Al., mnn-
minut Al.*/
  if(Serial.available()) {
    String cad=Serial.readString();
    if(cad.charAt(0)=='H') {
      h=cad.substring(1).toInt();
      if(h>23||h<0) h=0;
    }
    if(cad.charAt(0)=='M') {
      m=cad.substring(1).toInt();
      if(m>59||m<0) m=0;
    }
    if(cad.charAt(0)=='S') {
      s=cad.substring(1).toInt();
      if(s>59||s<0) s=0;
    }
    if(cad.charAt(0)=='h') {
      ha=cad.substring(1).toInt();
      if(ha>23||ha<0) ha=0;
    }
    if(cad.charAt(0)=='m') {
      ma=cad.substring(1).toInt();
      if(ma>59||ma<0) ma=0;
    }
  }
  //Activem, o no, l'alarma amb el polsador.
  if(digitalRead(2)==1) {
    activAl= !activAl;
    if(activAl==false) estatAl=0;
  }
  //Si l'alarma està activa i és l'hora, activem el so
en dos tons.
  if(activAl==true) {
    if(h==ha&&m==ma&&s==0) estatAl=1;
    if(estatAl==1) {tone(8,1000); estatAl=2;}
    else if(estatAl==2) {tone(8,50); estatAl=1;}
  }
  else noTone(8);

  //incrementem s, m, i/o h
  ++s;
  if(s==60) {
    ++m;
    s=0;
  }
  if(m==60) {
    ++h;
    m=0;
  }
  if(h==24) {
    h=0;
  }
  //Mostrem dades al LCD
  //lcd.clear();
```

lcd de la classe *LiquidCrystal*, indicant els pins on està connectat. També és declaren unes variables que han de ser d'àmbit global.

A la funció *setup()* s'inicialitzen els valors inicials per defecte de les variables globals i la connexió sèrie amb l'ordinador.

La funció *loop()* realitza les principals funcions en un bucle repetitiu.

Primer, comprova si s'ha rebut dades pel canal sèrie, si es així, s'interpreta i aplica la dada on correspongui (posar la hora, posar els minuts, posar l'alarma, etc.)

Posteriorment es comprova si s'ha polsat el polsador per si cal activar/desactivar l'alarma.

En cas que l'alarma estigui activada, es comprova si coincideixen l'hora actual i l'hora d'alarma per fer sonar el brunzidor, o no.

Seguidament es fa l'increment dels segons i, si s'escau, els minuts i les hores.

Un cop feta l'actualització de l'hora, es mostra al visualitzador LCD l'hora, l'hora d'alarma i, si de cas, la A indicadora que l'alarma està activada.

Si els dígits d'hora i d'alarma solament tenen un dígit, llavors s'afegeix davant el dígit 0.

```
  lcd.setCursor(0,0);
  lcd.print("Hora:   ");
  if(h<10) lcd.print("0");
  lcd.print(h);
  lcd.print(":");
  if(m<10) lcd.print("0");
  lcd.print(m);
  lcd.print(":");
  if(s<10) lcd.print("0");
  lcd.print(s);
  lcd.setCursor(0,1);
  lcd.print("Alarma: ");
  if(ha<10) lcd.print("0");
  lcd.print(ha);
  lcd.print(":");
  if(ma<10) lcd.print("0");
  lcd.print(ma);
  lcd.setCursor(14,1);
  if(activAl==true) lcd.print("A");
  else lcd.print(" ");
  if(estatAl!=0) lcd.print(estatAl);
  else lcd.print(" ");

  delay(1000);
}
```

Per acabar el bucle *loop()*, abans de tornar-se a repetir, fa una espera de 1 segon.

Encara que el bucle es repeteix cada 1 segon, a llarg plaç el rellotge no té precisió ja que cal tenir en compte el temps per fer totes les tasques d'atenció a la consola sèrie, gestió de l'alarma i visualització de dades que provoquen l'endarreriment. Aquesta imprecisió, però, no es té en compte per la intenció de fer que el codi sigui més fàcilment interpretable.

Una forma d'evitar la imprecisió del rellotge seria aplicar interrupcions[5] a temps precisos, mesurats per una base de temps externa, per fer el compteig de l'hora, aconseguint que sigui independent del temps emprat en la realització del loop().

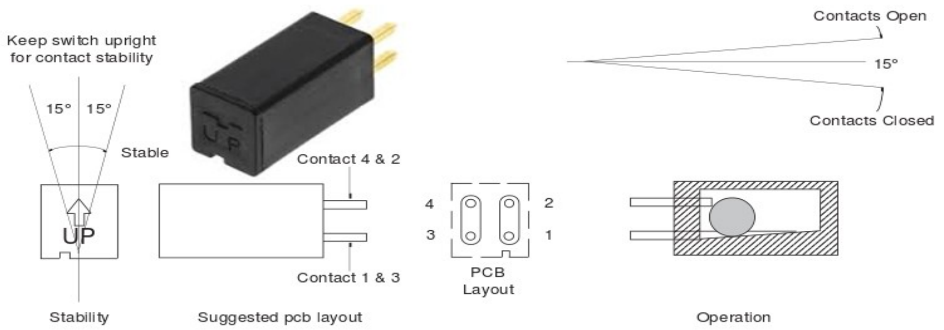
5.7.- Pràctica 7. Dau electrònic.

Implementem un circuit electrònic que simula la tirada d'un dau, mostrant, aleatòriament un resultat entre 1 i 6 representat mitjançant els punts de les cares del dau amb l'encesa d'uns led's.

Per efectuar la "tirada" podríem utilitzar un simple polsador, però, per fer-ho una mica més real, podem utilitzar un interruptor especial: un sensor d'inclinació "tilt switch" que connecta, o no, uns contactes en funció de l'angle d'inclinació o vibració de l'interruptor. Així, per efectuar una "tirada", haurem de moure enèrgicament el circuit amb la mà, observant com els leds es van il·luminant aleatòriament amb el moviment, i, en deixar el circuit immòbil, els leds s'aturaran mostrant un valor aleatori del dau.

Les característiques tècniques d'un interruptor especial, sensor d'inclinació o de vibració, són:

5 Les interrupcions són una tècnica dels sistemes microprocessadors de reclamar l'atenció del sistema per atendre d'immediat un esdeveniment produït que té una prioritat superior a la tasca que s'estigui desenvolupant actualment. Així, si produïm una interrupció per un clock de precisió extern, cada segon, llavors el comptatge del temps de rellotge serà més precís. Arduino Uno incorpora funcions específiques per fer el tractament d'interrupcions pels pins 2 i 3, com ara: attachInterrupt(), interrupts(), etc.

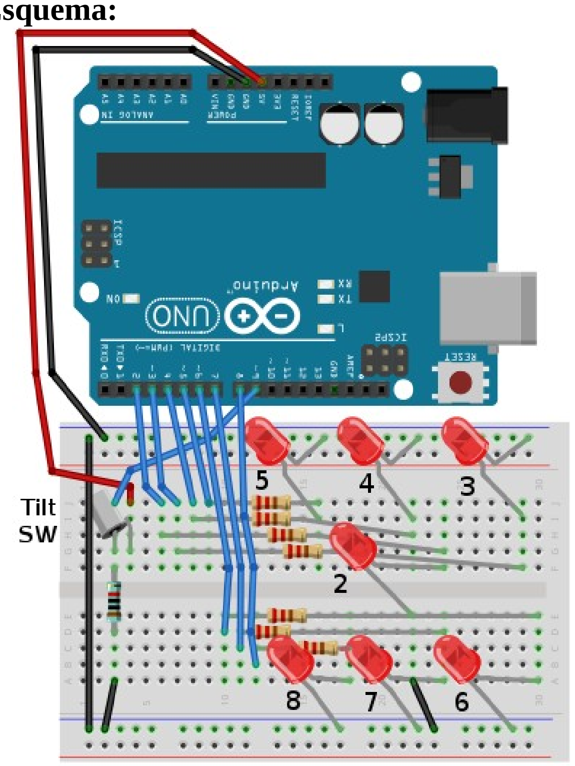


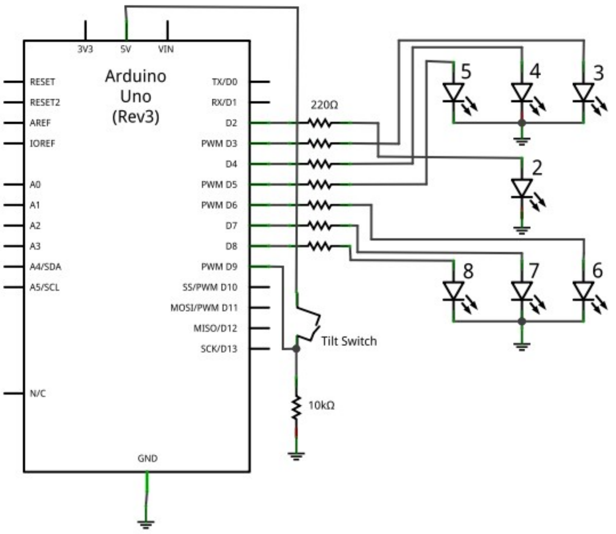
Interruptor especial d'inclinació RBS040100, 24v, 25mA, sensor de moviment, inclinació i vibració

El "tilt switch" consta de dos contactes que s'uneixen elèctricament mitjançant la posició d'una bola mòbil dins una càpsula. La bola es desplaçarà per efecte de la inclinació o de moviments bruscos.

Components: 7 leds, un sensor d'inclinació, 7 resistències de 220Ω i 1 resistència de 10kΩ.

Esquema:





Programa:

```
/*
  Pins de sortida on s'ha connectat els 7 leds que formen el dau:
  5 4 3
    2
  8 7 6

  Leds a il·luminar per a cada valor del dau:
```

El primer que es fa en el programa és definir de forma abstracta les sis cares del dau. Concretament

```
     ○ ○ ○   ● ○ ○   ● ○ ○   ● ○ ●   ● ○ ●   ● ● ●
       ●       ○       ●       ○       ●       ○
     ○ ○ ○   ○ ○ ●   ○ ○ ●   ● ○ ●   ● ○ ●   ● ● ●
     no. 1   no. 2   no. 3   no. 4   no. 5   no. 6
*/

/* Escriurem els leds en la seqüencia: 2,3,4,5,6,7,8.
  Arrays que indiquen, per cada numero, quin led s'il·lumina (1)
  i quin led s'apaga (0):
Sortides:          2 3 4 5 6 7 8 */
int cara1Dau[7]={ 1,0,0,0,0,0,0 }; //no. 1
int cara2Dau[7]={ 0,0,0,1,1,0,0 }; //no. 2
int cara3Dau[7]={ 1,0,0,1,1,0,0 }; //no. 3
int cara4Dau[7]={ 0,1,0,1,1,0,1 }; //no. 4
int cara5Dau[7]={ 1,1,0,1,1,0,1 }; //no. 5
int cara6Dau[7]={ 0,1,1,1,1,1,1 }; //no. 6

void setup() {
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,INPUT); //Tilt switch

  //Inicialitzem la llavor aleatòria (llegim un valor de "soroll"
  //des de l'entrada analògica A6 que està desconnectada.
  randomSeed(analogRead(A6));
}

void loop() {
  int n;
  if(digitalRead(9)==LOW) {
    n=random(1, 7);
    posaNumeroDau(n);
  }
}

void posaNumeroDau(int val) {
  //Iterem, de sortida 2 a 8, i per cada valor de "a"
  //posem el que indica l'array corresponent al
  //valor a representar.
  for(int a=2; a<=8; ++a) {
   switch (val) {
     case 1:
       digitalWrite(a, cara1Dau[a-2]);
       break;
     case 2:
       digitalWrite(a, cara2Dau[a-2]);
       break;
     case 3:
       digitalWrite(a, cara3Dau[a-2]);
       break;
     case 4:
       digitalWrite(a, cara4Dau[a-2]);
       break;
     case 5:
       digitalWrite(a, cara5Dau[a-2]);
       break;
     case 6:
       digitalWrite(a, cara6Dau[a-2]);
```

s'implementa en uns arrays o vectors (un per cada cara), l'estat de cada led per representar el número: 1 és encès, 0 és apagat.

Dins el bloc *setup()* inicialitzem el mode de cada pin i definim la llavor del generador de números aleatoris amb *randomSeed()*. Això ho fem agafan un valor aleatori provinent del "soroll" (noise) que es llegeix d'una entrada analògica que està desconnectada.

Dins de loop() comprovem constantment si l'entrada del contacte sensor d'inclinació està a nivell baix, si es així, generem un número aleatori entre 1 i 6, ambdós inclosos i el representem amb els leds mitjançant la funció posaNumeroDau().

La funció posaNumeroDau() s'ha creat per encarregar-se de il·luminar els leds segons el número indicat. Per això, segons sigui el cas (1,2,...6) recorre els pins de sortida amb el for i l'il·lumina segons s'indica a l'array corresponent al número.

```
        break;
    }
  }
}
```

5.8.- Pràctica 8. Control de potència.

Els circuits elèctrics de potència treballen amb tensions de l'ordre de centenars de volts i intensitats de l'ordre de desenes d'ampers. Aquests valors són molt més elevats que els que se solen utilitzar en circuits electrònics de control. Això fa que no es puguin aplicar ni controlar directament els circuits de potència des dels circuits electrònics de control, com és el cas d'Arduino. És a dir, activar un llum de 100w a 230v de c.a., o un motor de centenars de vats a tensions monofàsiques o trifàsiques, no és el mateix, ni de lluny, que il·luminar un Led, o fer sonar un brunzidor o similar.

Però, això no vol dir que no es pugui fer. Primer que tot cal assegurar-nos que els circuits de tensió i intensitat elevada no arribin a interaccionar elèctricament mai directament amb les tensions i intensitats de baix valor que utilitzen els circuits electrònics de control. Generalment, aquesta separació elèctrica de circuits se sol aconseguir mitjançant acoblaments mecànics (cas del relé o contactor) o mitjançant acoblaments òptics (cas dels optoacobladors).

En aquesta pràctica veurem els principals sistemes d'acoblament entre circuits de potència i circuits de control mantenint la separació elèctrica dels dos circuits.

Components: 1 diode 1N4004, 1 relé 5v, 1 TRT NPN BC547, 1 optoacoblador MOC3040, 1 triac TIC226, 1 optoacoblador H11AA1, 2 polsadors, 1 condensador de 10nF, 1 condensador de 150nF, 4 resistors de 10KΩ, 1 resistor 220Ω, 1 resistor de 56Ω, 1 resistor de 39Ω, 1 resistor de 330Ω, 1 resistor 910Ω, 1 resistor de 820Ω i 1 resistor de 680kΩ.

El **circuit de sortida de control amb relé** controla l'encesa d'un llum de potència (P1) connectat a la xarxa elèctrica de 230v. La bobina del relé es activada per un transistor que, a la vegada, és activat per la sortida 2 de Arduino. En paral·lel amb la bobina tenim el diode volant.

El **circuit de sortida de control amb un triac**[6] també controla l'encesa d'un llum de potència (P2) que està connectat a la xarxa elèctrica de 230v. L'aïllament elèctric s'aconsegueix mitjançant l'optoacoblador MOC3040 que acobla òpticament un led i un fotodiac[7] amb detecció de pas per zero de la c.a., per reduir la producció d'interferències electromagnètiques. El led de l'optoacoblador s'activa des de la sortida 3 de Arduino.

El **circuit d'entrada optoacoblada** consisteix en una adaptació de la tensió de 230v de la xarxa elèctrica per aplicar-la a una entrada de Arduino. D'aquesta forma, es pot tenir una entrada del sistema que indicarà l'existència, o no, de la tensió de 230v de c.a. Aquesta està acoblada mitjançant un optoacoblador format per dos leds en antiparal·lel (s'il·lumina un o altre, segons sigui el

6 Un triac és un dispositiu semiconductor format per dos tiristors en antiparal·lel amb porta comuna. Quan s'aplica senyal a la porta, entra en conducció permetent el pas del corrent de potència en els dos sentits (per a c.a.). És pot considerar al triac com un interruptor estàtic de potència per a circuits de c.a. controlat per la porta (gate).

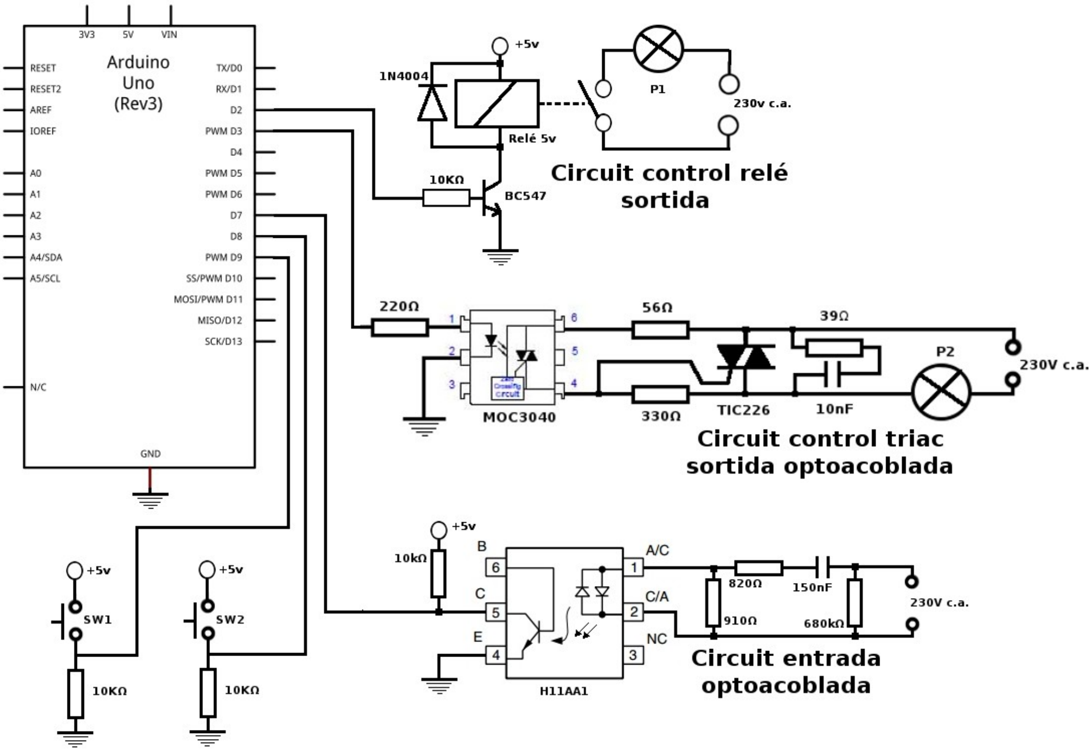
7 El diac és un dispositiu semiconductor utilitzat en exclusiva per l'encebat de triacs, el diac entra en conducció quan la tensió en borns supera els 30v aproximadament, en els dos sentits. En el cas del fotodiac, aquest també entra en conducció quan incideix la llum del led sobre ell.

semiperíode de l'ona de c.a.) que il·luminen un fototransistor que conduirà segons li arribi llum dels leds. El circuit està connectat a l'entrada 7 de Arduino i és de lògica negativa, es a dir, quan hi ha aplicada una tensió de 230 v c.a. subministrarà un "0" a l'entrada i, en cas contrari, subministrarà un "1".

La utilitat del circuit d'entrada acoblada no es solament per saber si tenim tensió de c.a. aplicada. També pot servir com a **detector de pas per zero** de la ona de c.a. Aquesta detecció ens pot subministrar un seguit d'impulsos, corresponents a cada pas per zero, que es pot utilitzar com a base de temps, amb la precisió que ens ofereix la freqüència de la xarxa elèctrica, per a rellotges, temporitzadors, etc. que ho requereixin.

En el circuit també tenim dos polsadors (SW1 i SW2) que són utilitzats per activar la sortida amb relé o la sortida amb triac, respectivament.

Esquema:



Programa:

```
#define P1 2
#define P2 3
#define IAC 7
#define SW1 9
#define SW2 8

void setup() {
  pinMode(P1, OUTPUT);
  pinMode(P2, OUTPUT);
  pinMode(IAC, INPUT);
  pinMode(SW1, INPUT);
```

El programa comença definint noms per als pins de sortida i d'entrada.

Dins de *setup()* es configura el mode entrada o sortida dels pins connectats.

El bloc *loop()* repeteix contínuament la lectura de les

```
  pinMode(SW2, INPUT);
}

void loop() {
  if(digitalRead(SW1)==HIGH ||
     !digitalRead(IAC)==HIGH) digitalWrite(P1, HIGH);
  else digitalWrite(P1, LOW);
  if(digitalRead(SW2)==HIGH) digitalWrite(P2, HIGH);
  else digitalWrite(P2, LOW);
}
```

entrades i, segons valguin, activa la sortida corresponent.
Si està activat el SW1 o està activat l'entrada de c.a., s'activarà P1, sinó, es desactivarà (cal notar la negació: **!**, de l'entrada de c.a.).

Si està activat SW2, s'activarà P2.

Pel control de circuits de potència elèctrica elevada, mitjançant circuits electrònics de control, cal tenir en compte una afectació important: els efectes de les pertorbacions electromagnètiques produïdes pels corrents elèctrics forts, principalment originats per càrregues inductives. Aquestes pertorbacions electromagnètiques poden provocar interferències en el control electrònic fent que funcioni malament.

Per protegir aquests circuits d'aquestes pertorbacions, existeixen tècniques que redueixen, o eliminen, el seu efecte, mitjançant filtres amb bobines i condensadors aplicats adequadament. En els esquemes utilitzats en aquesta pràctica no s'ha aplicat cap d'aquestes tècniques de protecció. El seu estudi requereix un major aprofundiment en el tema que no tenim en aquest dossier. Per trobar més informació i aplicació d'aquestes tècniques de protecció electromagnètica, cal cercar-la en altres manuals més especialitzats.

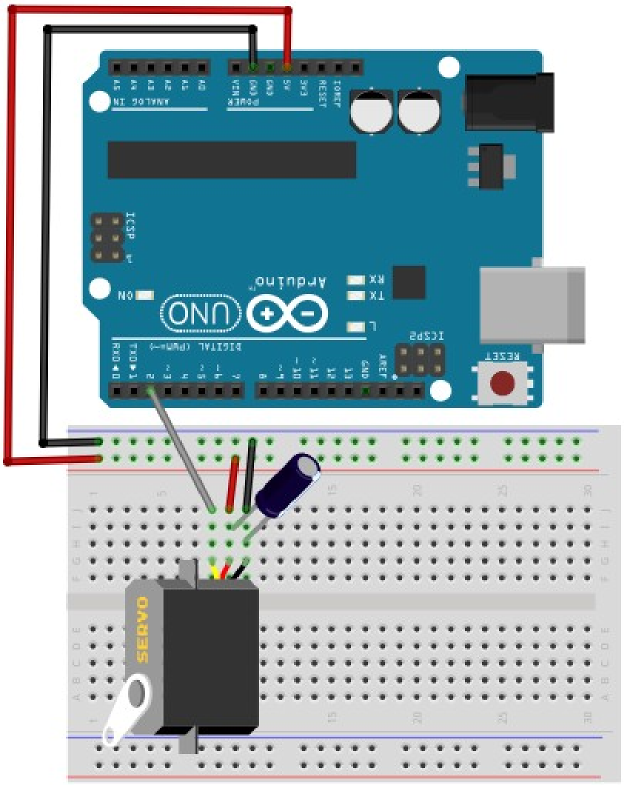
5.9.- Pràctica 9. Control d'un servomotor.

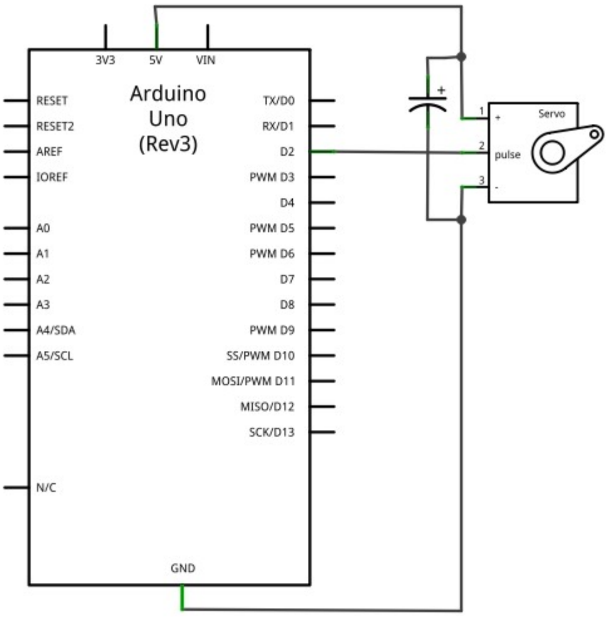
Ajustar l'angle de posició d'un servomotor SM-S2309S per que canviï alternativament als angles a: 0º – 60º – 120º – 60º, periòdicament.

Components: 1 servomotor SM-S2309S, 1 condensador electrolític de 100µF.

Per determinar l'angle de posicionament del servo ho fem amb l'aplicació d'un senyal PWM, on el valor analògic aplicat indicarà un angle concret. Experimentalment s'ha observat que els valors PWM entre els extrems de posicionament de l'eix són entre 0 i 174 que corresponen als angles entre 0º i 120º en el servo SM-S2309S. Qualsevol altre angle entre els límits indicats serà un valor proporcional al corresponent entre els rang numèric de 0 a 174, així, si volem posicionar l'angle a 22º, correspondria un valor PWM de: $22 \times \frac{(174-0)}{(120-0)} \approx 32$, aquesta operació és un mapejat de dades de un rang entre 120 i 0 i un altre rang entre 174 i 0. El valor 22 en el rang 120 i 0 correspon al valor 32 en el rang 174 i 0. Aquesta operació de mapejat esta implementada en una funció matemàtica del llenguatge de programació Arduino, concretament la funció *map* amb la següent sintaxis: **map(valor, fromLow, fromHigh, toLow, toHigh).** En l'exemple previ correspondria a: *map(22, 0, 120, 0, 174);* i retornaria el resultat: 32.

Esquema:





Programa:

```
//Afegim la llibreria Servo.h que introdueix les
//instruccions de control de servomotors.
#include <Servo.h>
//Creem un objecte Servo:
Servo svmotor;

void setup() {
  svmotor.attach(2); //el control és al pin 2
}

void loop() {
  /*El model de servo SM-S2309 gira 120° (+-60°),
on 0° correspon a un valor PWM de 0 i 120°
correspon a un valor PWM de 174, la resta d'angles
són proporcionals.*/
  int t=1200;
  int anglemin=0; //->0°
  int anglemax=174; //->120°

  svmotor.write(anglemin); //0°
  delay(t);
  svmotor.write(anglemax/2); //60°
  delay(t);
  svmotor.write(anglemax); //120°
  delay(t);
  svmotor.write(anglemax/2); //60°
  delay(t);
}
```

Per utilitzar el servomotor hem de carregar la llibreria *Servo.h* que ens proveeix d'un conjunt de noves funcions específiques pel control d'aquests dispositius.

La classe *Servo* serveix per crear objectes tipus *Servo*. Cada objecte proveeix d'un conjunt de propietats i mètodes pel control.

Amb *attach* indiquem a quin pin està connectat el born de senyal PWM del servo.
Posteriorment ja podem indicar al Servo quina posició volem amb el mètode *write(val)*, on *val* és el valor PWM corresponent al angle.
El programa posiciona els angles 0º, 60º, 120º i 60º seqüencialment i repetidament.

5.10.- Pràctica 10. Control motor c.c.

Creació d'un circuit i programa per controlar dos motors de c.c.

El Motor 1 estarà controlat manualment per un potenciòmetre, que en girar de la meitat cap a l'esquerra farà girar el motor a l'esquerra amb velocitat variable segons la posició del potenciòmetre i en girar de la meitat cap a la dreta, farà girar el motor a la dreta, també amb velocitat variable. El control es farà mitjançant un pont en H amb el I.C. L293D.

El Motor 2 tindrà un control més simple, podrà regular la seva velocitat però, no podrà invertir el gir. El control es farà mitjançant un transistor Mosfet de potència. Mitjançant el codi del programa farem que s'engegui a màxima velocitat i s'aturi 5 vegades, després, farem que acceleri i desacceleri lentament des de zero fins a màxima velocitat i viceversa, repetint tot el procés periòdicament.

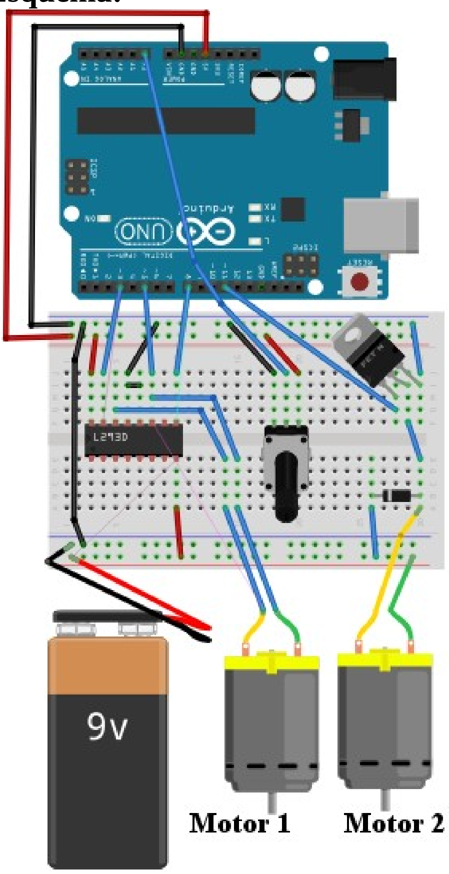
S'utilitzarà la consola sèrie per rebre informació de l'evolució del programa i veure els valor de les principals variables.

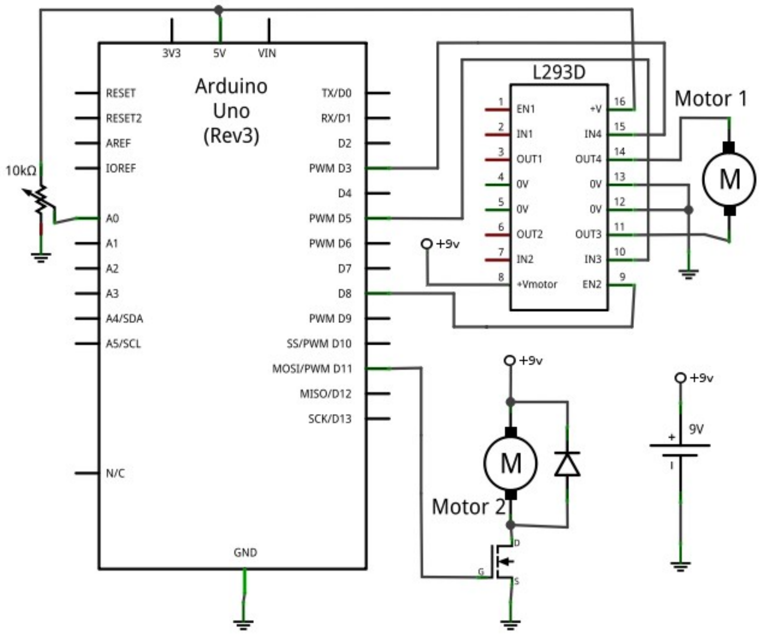
Alimentació:
El circuit Arduino i els components connectats s'alimenten, per defecte, directament des del cable USB que connecta a l'ordinador, però, si el consum elèctric de la càrrega és elevat (com es el cas dels motors de c.c.) llavors cal aportar una tensió addicional per aplicar al mateix motor. Per tant, en aquest circuit afegim una pila de 9v que subministrarà el voltatge i corrent suficient per que funcioni els dos motors de c.c. La resta del circuit continuarà alimentant-se des de la tensió de 5v que li subministra el connector USB, igual com s'ha fet fins ara.
Es pot observar a l'esquema com el positiu de la pila de 9v alimenta directament un born del Motor 2 i el pin de la tensió del Motor 1. El born negatiu de la pila es connecta al negatiu de la resta del circuit, per tant, tenim una part del circuit a 9v i una altra part a 5v i comparteixen el negatiu del circuit.

En tots dos casos, la variació de velocitat s'aconsegueix aplicant impulsos PWM al control del motor.
Per al Motor 1, la inversió de gir s'aconsegueix segons quin sigui el voltatge (mitjana del valor PWM) aplicat a cada born del motor, els dos voltatges han de ser complementaris entre ells, per tant, a un born se li aplica el valor PWM corresponent i a l'altre, el valor PWM complementari, calculat com la diferència de 255 i el valor del primer. Aquesta diferència pot ser positiva, negativa o zero. En els dos primer casos fa girar el motor a dreta o esquerra i quan és zero (coincidint amb la posició central del potenciòmetre), el motor estarà aturat.

Components: 2 motors de c.c., 1 I.C. L293D, 1 transistor Mosfet IRF520, 1 diode 1N4007, 1 potenciòmetre de 10kΩ, 1 portapiles 9v, 1 pila 9v.

Esquema:





Programa:

```
void setup() {
  pinMode(11,OUTPUT);
  pinMode(13, OUTPUT);

  pinMode(3, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(8, OUTPUT);

  digitalWrite(8, HIGH); //habilitem Enable de motor 1.
  Serial.begin(9600);
}

void loop() {
  //Motor 1:
  /*int v, vv, pot;
  Serial.print("Motor 1, Potenciometre: ");
  pot=analogRead(A0); //lectura pot. entre 0 i 1023
  Serial.print(pot);
  v=map(pot,0,1023,0,255); //Mapejat de rang
  vv=255-v; //valor complementari de v
  if(v>vv) {
    Serial.print(" dreta: ");
    Serial.println(v);
  }
  else {
    Serial.print(" esquerra: ");
```

El programa consta de dos blocs de codi, un per a cada motor i no poden funcionar els dos simultàniament, es a dir, cal provar el funcionament de cada motor per separat, per això hem de comentar (afegir: /* */) i descomentar (treure: /* */) el codi corresponent.

Per provar el codi del Motor 1, es comenta el codi corresponent al Motor 2 i es descomenta el codi corresponent al Motor 1 i es prova el funcionament del Motor 1.

Posteriorment, per provar el codi del Motor 2, es comenten i descomenten els blocs de codi al

```
    Serial.println(vv);
  }
  analogWrite(3, v);
  analogWrite(5,vv);*/

  //Motor 2:
  /*Serial.println("Motor 2, atura i engega 5 cops.");
  for(int a=0; a<5; ++a) {
    digitalWrite(11, HIGH);
    delay(1500);
    digitalWrite(11, LOW);
    delay(1500);
  }
  Serial.println("Accelera i desaccelera
progressivament.");
  for(int a=0; a<255; ++a){
    analogWrite(11,a);
    delay(50);
  }
  for(int a=255; a>0; --a){
    analogWrite(11,a);
    delay(50);
  }*/
}
```

contrari i es prova el funcionament del Motor 2.

En cada cas, per la consola sèrie es pot llegir missatges informatius de l'evolució del programa i les seves variables.

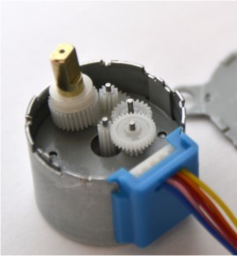
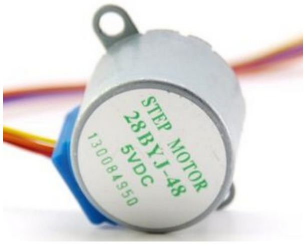
5.11.- Pràctica 11. Control d'un motor pas a pas.

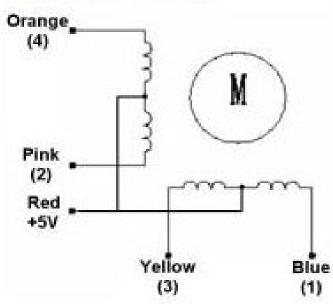
Mitjançant un circuit de control de motors pas a pas, implementat amb el I.C. L293D, volem controlar el gir d'un motor pas a pas. Els passos de gir s'indicaran interactivament per la consola sèrie, on indicarem el nombre de passos a girar i el sentir de gir, dreta o esquerra. El motor pas a pas, s'ha de desplaçar els passos indicats per la consola sèrie, en el sentit corresponent.

Per implementar la pràctica utilitzarem el xip L293D i un motor pas a pas de 5v, model 28BYJ-48.

Característiques tècniques del motor pas a pas 28BYJ-48 5v:
- Motor de de 4 fases (2 fases amb presa intermitja).
- El motor porta uns engranatges reductors de velocitat amb relació aproximada de 1:64
- Angle de pas: del motor intern: 5'625º (64 passos per volta) (Seqüència de 8 passos, half step)
  del motor intern: 11'25º (32 passos per volta) (Seqüència de 4 passos, full step)
  (la llibreria d'Arduino Stepper.h esta configurada per treballa en full step a 11'25º)

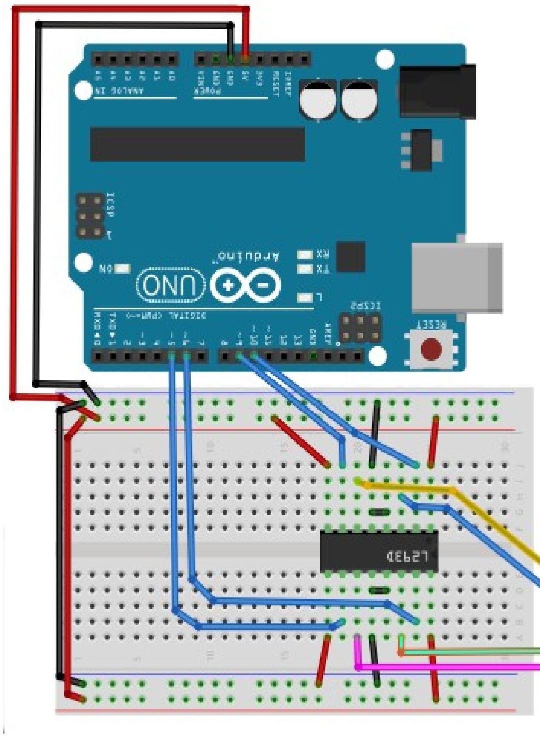
Per tant, treballant en el mode de 11'25º per pas, que corresponen a 32 passos per fer una volta de motor (32x11'25º=360º), realment caldrà aplicar 32x64=2048 impulsos per que l'eix de sortida faci una volta sencera.

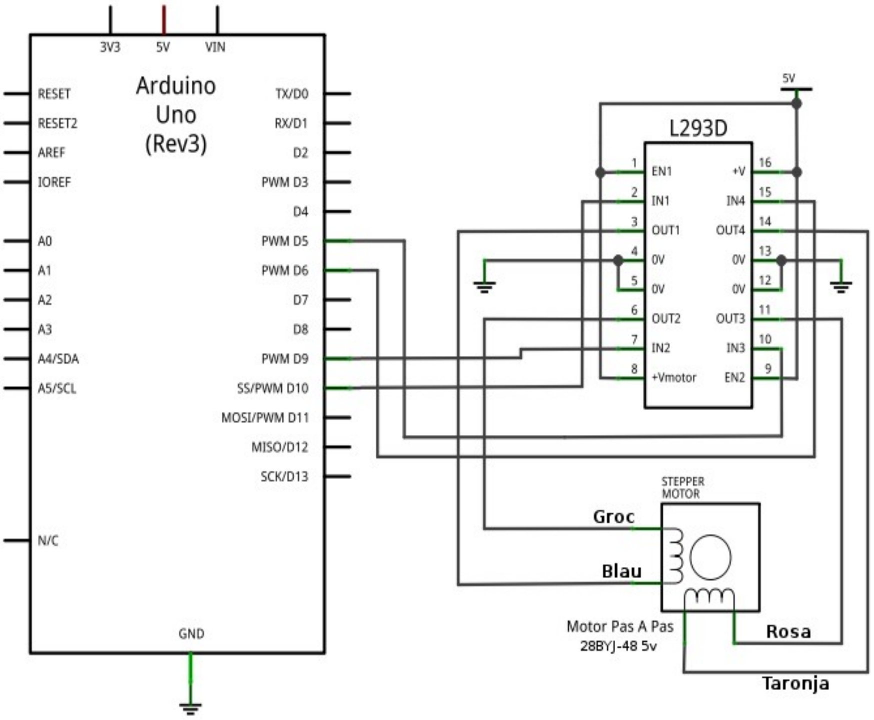





Components: 1 Motor pas a pas model 28BYJ-48 – 5V, 1 I.C. L293D

Esquema:





En el motor utilitzat, existeix un cinquè cable de connexió de color vermell que correspon a la presa intermitja dels debanats per al seu ús com a motor pas a pas unipolar. En aquesta pràctica usem el motor com a bipolar, per tant, el cable vermell no es connecta enlloc.

Nota: L'esquema implementat pel control del L293D, utilitza 4 pins de connexió amb Arduino (pins 5, 6 i 9, 10, emparellats), però, si es vol, també es pot implementar la opció de connexió a Arduino amb 2 pins, afegint al circuit dos TRT que inverteixen la lògica. Aquest altre esquema que connecta amb 2 pins, es pot veure representat a l'apartat 3.10 del control del motor pas a pas d'aquest mateix dossier.

A efectes pràctics, el funcionament amb 4 pins o 2 pins de connexió, és el mateix, ja que la senyal de control a cada parella de pins de control és la mateixa però invertida a cada pin, i, aquesta inversió la podem fer per programa, com es el cas de usar 4 pins o, la podem fer electrònicament, amb l'ajut de dos transistors inversors, com es el cas de usar 2 pins. En aquesta situació de usar 2 pins, també cal canviar la definició de l'objecte Stepper al principi del programa de control, indicant solament els dos pins connectats.

Programa:

```
#include <Stepper.h>

// pins on esta connectat el control
// del motor PaP
#define pap1 10
#define pap2 9
#define pap3 5
#define pap4 6

//Creem l'objecte motorpap de la
//classe Stepper:
Stepper motorpap(2048, pap1, pap2, pap3, pap4);

void setup() {
 pinMode(pap1, OUTPUT);
 pinMode(pap2, OUTPUT);
 pinMode(pap3, OUTPUT);
 pinMode(pap4, OUTPUT);

 // provem el gir fent una volta a
 // dreta i esquerra a la velocitat de
 // 60 rpm.
 motorpap.setSpeed(60);
 motorpap.step(-2048);
 motorpap.step(2048);


 motorpap.setSpeed(40);
 Serial.begin(9600);

}
```

Iniciarem el programa incloent la llibreria *Stepper.h* que afegeix les funcions específiques del control del motor pas a pas.
Definim els pins connectats al circuit de control amb L293D del motor.

També declarem un objecte de nom *motorpap* de la classe *Stepper*, indicant el nombre de passos per fer una volta i els pins on està connectat el control.

Dins de *setup()* definim els pins del control del motor PaP com a sortida.

També definim la velocitat de gir (en rpm) i fem que l'eix del motor faci una volta sencera cap a la dreta i una altra cap a l'esquerra (en aquest model de motor, treballant a *full step* i tenint en compte la relació del reductor, correspon 2048 passos per fer una volta completa).

```
void loop() {
 // Llegim els passos a girar de la consola
 // sèrie. Si és un valor positiu gira en
 // un sentit, si és negatiu, gira en
 // l'altre sentit
 if(Serial.available()) {
  int passos=Serial.parseInt();
  motorpap.step(passos);
  Serial.println(passos);
 }
}
```

Un cop feta la inicialització, dins de *loop()*, llegirem contínuament si s'indica passos a girar mitjançant la inserció numèrica per la consola sèrie. Si posem valors numèrics positius girarà els passos indicats en un sentit i si indiquem valors numèrics negatius, farà els passos en sentit contrari.

En el funcionament de la pràctica amb el motor pas a pas, es pot observar que quan el motor està aturat, l'eix del motor no es pot fer girar manualment per què esta blocat, ja que, els debanats són alimentats igualment i s'està aplicant un fre electromagnètic.

5.12.- Pràctica 12. Mòbil. Plataforma i control motors.

En aquesta pràctica és pretén implementar una plataforma mòbil controlada per Arduino. Mitjançant dos motors c.c., amb reductor de velocitat incorporat, farem girar dues rodes motrius. La tercera roda de suport de la plataforma serà una roda boja.

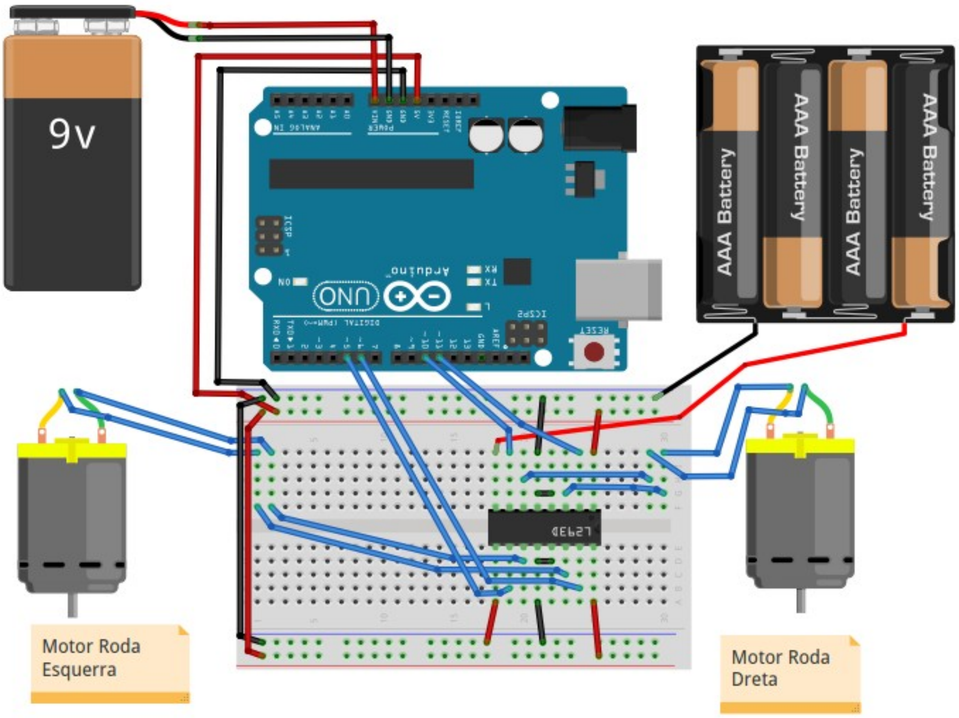
La plataforma mòbil implementada ha de servir de base per a altres pràctiques posteriors on s'afegirà altres elements de control del mòbil.

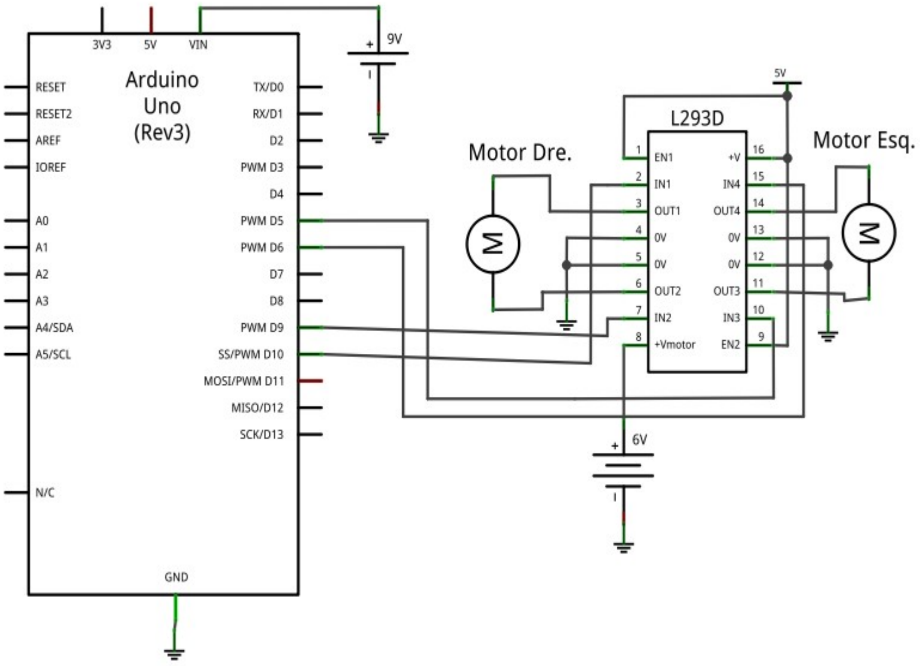
Per al control dels motors s'implementarà unes funcions que s'encarreguen de moure els motors adequadament per a fer moviments de desplaçament: *desplacament(), rodaD() i rodaE()*.

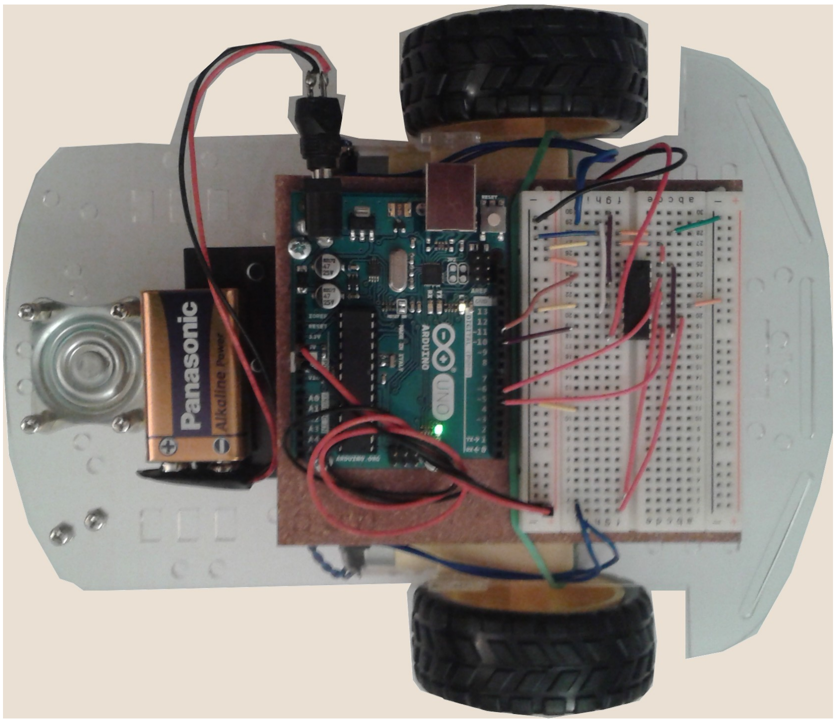
Per provar les funcions de moviment dels motors i desplaçament del mòbil, s'implementarà un seguit de moviments de gir a dreta i esquerra, i avanç i retrocés del mòbil.

Components: Xassís mòbil de dues rodes motrius amb motor-reductor de c.c. i roda boja, bateria 9v, bateries 6v, I.C. L293D

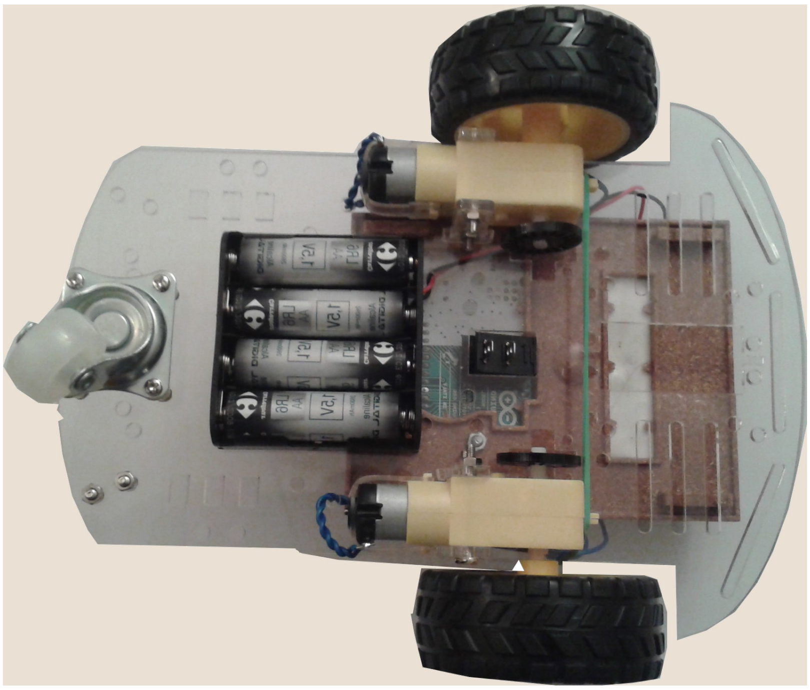
Esquema:


9v
AAA Battery
AAA Battery
AAA Battery
AAA Battery
L293D
Motor Roda
Esquerra
Motor Roda
Dreta


3V3
5V
VIN
9V
Arduino
Uno
(Rev3)
RESET
RESET2
AREF
IOREF
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
GND
TX/D0
RX/D1
D2
PWM D3
D4
PWM D5
PWM D6
D7
D8
PWM D9
SS/PWM D10
MOSI/PWM D11
MISO/D12
SCK/D13
5V
L293D
Motor Dre.
Motor Esq.
EN1
IN1
OUT1
0V
0V
OUT2
IN2
+Vmotor
+V
IN4
OUT4
0V
0V
OUT3
IN3
EN2
6V



Vista superior del xassís mòbil amb dues rodes motrius i una roda boja, amb la placa Arduino i protoboard ubicada sobre la plataforma i connectada a una bateria d'alimentació i als cables de connexió amb els motors controlats per l'I.C. L293D.



Vista inferior del xassís mòbil on es veu els dos motors c.c. amb reductor, acoblats a les rodes motrius i les bateries d'alimentació exclusiva dels motors de c.c.

Programa:

```
#define aRE 10  //pin avanç roda esquerra
#define rRE 9  //pin retrocés roda esquerra
#define aRD 6   //pin avanç roda dreta
#define rRD 5   //pin retrocés roda dreta

void desplacament(int v=0, char sentit='a', char gir='n', int
radi=0);

void setup() {
  pinMode(aRE, OUTPUT);
  pinMode(rRE, OUTPUT);
  pinMode(aRD, OUTPUT);
  pinMode(rRD, OUTPUT);

  //Moviments del mòbil:
  int t=2000, v=40;
  desplacament(80); //endavant recte a un 80% de velocitat
  delay(t);
  desplacament(v,'a', 'e', 0); //endavant gir tancat a
esquerra
  delay(t);
  desplacament(v,'a', 'd', 70); //endavant gir obert a dreta
  delay(t);
  desplacament(v,'r', 'n', 0); //enrere recte
  delay(t);
  desplacament(v,'r', 'e', 60); //enrere gir obert a esquerra
  delay(t);
  desplacament(v,'r', 'd', 0); //enrere gir tancat a dreta
  delay(t);
  desplacament(v,'a', 'd', 101); //gir dreta sobre el seu eix
  delay(t);
  desplacament(v,'a', 'e', 101); //gir esquerra sobre el seu
eix
  delay(t);
  desplacament(0); //aturada
}

void loop() {
  //no es fa res repetitiu.
}

void desplacament(int v, char sentit, char gir, int radi) {
  //v: vel. en % 0-100, sentit:(a o r),
  //gir: (d,e,n), radi: radi gir 0-100 o 101

  //validem els paràmetres:
  if(v>100) v=100; if(v<0) v=0;
  if(radi>100&&radi!=101) radi=100; if(radi<0) radi=0;
  if(sentit!='a'&&sentit!='r') sentit='a';
  if(gir!='d'&&gir!='e'&&gir!='n') gir='n';

  if(gir=='n') { //sense gir
    rodaD(v, sentit);
    rodaE(v, sentit);
  }
  if(radi==101) { //gir sobre el seu propi eix
    if(gir=='d') {
      rodaD(v, 'r');
      rodaE(v, 'a');
    }
    if(gir=='e') {
```

Definim els pins on estan connectats l'avanç i retrocés de cada motor.

Declarem la funció *desplacament()*, indicant els seus paràmetres i els seus valors per defecte, en cas que no s'indiqui el paràmetre.

Inicialitzem el mode de cada pin com a sortida.

Fem moure el mòbil endavant, enrere, girant amb radis de gir tancats o oberts o girant sobre si mateix a dreta i esquerra.

Tota l'activitat esta dins la funció setup(), per tant, solament es farà una sola vegada, en acabar, s'aturarà i caldrà fer reset per repetir-ho.

Dins de la funció loop() no hi fem res.

La funció *desplacament()* està dissenyada per encarregar-se de coordinar el gir de les dos rodes adequadament per fer els moviments desitjats segons s'indica en el valor dels paràmetres.
Podem indicar el % de velocitat d'avanç, retrocés o gir, el sentit de desplaçament, el gir i el radi de gir o gir sobre el seu eix.

```
      rodaD(v, 'a');
      rodaE(v, 'r');
    }
  }
  else { //gir amb radi
    int rg=map(radi,0,100,0,v);
    if(gir=='d') {
      rodaD(v, sentit);
      rodaE(rg, sentit);
    }
    if(gir=='e') {
      rodaD(rg, sentit);
      rodaE(v, sentit);
    }
  }
}

void rodaD(int v, char sentit) {
   //fa girar la roda dreta endavant o endarrere
   //a la velocitat percentual indicada a v
   // v: velocitat en %, sentit: (a o r)
   int vd=255*v/100;
   if(sentit=='a') {
     analogWrite(aRD, vd);
     analogWrite(rRD,0);
   } else {
     analogWrite(rRD, vd);
     analogWrite(aRD,0);
   }
}

void rodaE(int v, char sentit) {
   //fa girar la roda esquerra endavant o endarrere
   //a la velocitat percentual indicada a v
   // v: velocitat en %, sentit: (a o r)
   int ve=255*v/100;
   if(sentit=='a') {
     analogWrite(aRE, ve);
     analogWrite(rRE,0);
   } else {
     analogWrite(rRE, ve);
     analogWrite(aRE,0);
   }
}
```

Les funcions *rodaD()* i *rodaE()* s'encarreguen de moure la roda corresponent segons els paràmetres que se li indiquen de velocitat i sentit.

Les funcions implementades: *desplacament(), rodaE() i rodaD()* són les que s'utilitzaran per moure el mòbil a partir de les dades obtingudes de sensors que s'aplicaran en pràctiques posteriors. Si en provar el funcionament, alguna roda gira en sentit contrari del que s'espera, cal invertir les connexions dels cables del motor corresponent.

Cal evitar els valors de velocitat molt baixa en els motors, ja que la seva pròpia inèrcia fa que el moviment no s'iniciï fins superar un valor mínim. Aquest fet i altres, provoquen imprecisions en els moviments del mòbil ja que els motors no estan realimentats, és a dir, els dos motors no tenen per que comportar-se igual en resposta a una determinada tensió aplicada igual per tots dos, encara que siguin de la mateixa marca i model.

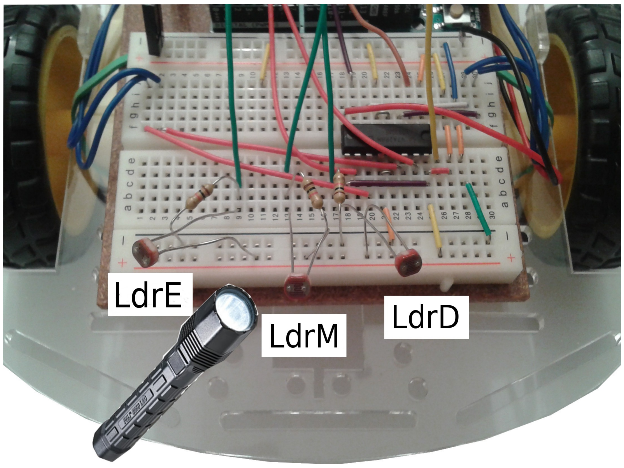
5.13.- Pràctica 13. Mòbil. Seguidor de llum.

A partir de l'estructura, circuit i programa de control de motors de la plataforma mòbil de la pràctica 12, implementarem, addicionalment, uns sensors de llum Ldr en el mòbil, que li permetin seguir una llum, tipus llanterna, orientada cap a les fotocèl·lules, en línia recta o girant a dreta o esquerra, segons calgui.

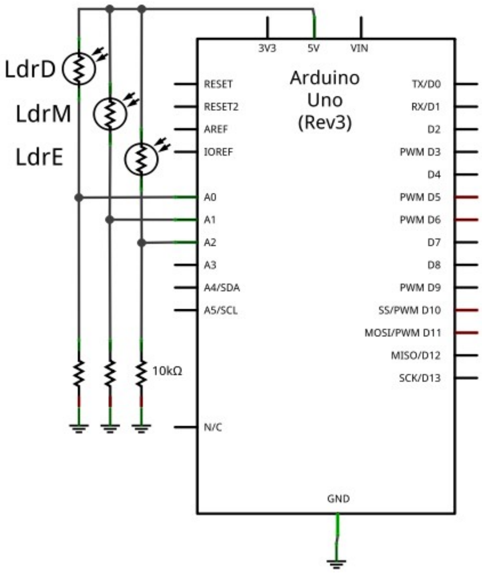
Per implementar aquest seguidor de llum, utilitzarem tres Ldr's ubicades a la part frontal del mòbil. La Ldr central orientada al mig, i les altres dos, orientades lleugerament a la dreta i l'esquerra.
Quan la Ldr del mig rebi una quantitat de llum més elevada que la ambiental i superior a la que reben les altres dos Ldr's, el mòbil es desplaçarà recte endavant, quan la Ldr de la dreta rebi una llum més elevada que la ambiental i més gran que la de les altres, el mòbil es desplaçarà girant a l'esquerra, i, quan la Ldr de l'esquerra rebi una llum més elevada que la ambiental i més gran que la que reben les altres, el mòbil es desplaçarà girant a la dreta.

Components: A més del components de la plataforma mòbil de la pràctica 12, 3 resistors LDR, 3 resistors de 10kΩ.

Esquema:



Plataforma mòbil on s'ha afegit les tres LDR a la part frontal que s'han il·luminar amb una llanterna o similar per controlar el moviment del mòbil.



Circuit de sensors LDR afegits al circuit de la plataforma mòbil de la pràctica 12.

Programa:

```
#define aRE 10  //pin avanç roda esquerra
#define rRE 9  //pin retrocés roda esquerra
#define aRD 6   //pin avanç roda dreta
#define rRD 5   //pin retrocés roda dreta

#define LdrD A0
#define LdrM A1
#define LdrE A2

void desplacament(int v=0, char sentit='a', char gir='n',
int radi=0);
```

Definim els pins analògics on es rep la tensió provinent de cada Ldr.

Declarem la funció *desplacament()* i els seus valors per defecte.

```
void setup() {
  pinMode(aRE, OUTPUT);
  pinMode(rRE, OUTPUT);
  pinMode(aRD, OUTPUT);
  pinMode(rRD, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  int sensLdrD=analogRead(LdrD);
  int sensLdrM=analogRead(LdrM);
  int sensLdrE=analogRead(LdrE);

  Serial.print(sensLdrE);
  Serial.print(" ");
  Serial.print(sensLdrM);
  Serial.print(" ");
  Serial.println(sensLdrD);

  int v=80, llindar=500; //cal ajustar el valor de la
variable llindar a la situació real de la prova!
  if(sensLdrD>sensLdrM && sensLdrD>llindar)
desplacament(v,'a', 'e', 30);
  else if(sensLdrE>sensLdrM && sensLdrE>llindar)
desplacament(v,'a', 'd', 30);
  else if(sensLdrM>llindar) desplacament(v,'a', 'n', 0);
  else desplacament(0);
}

//definir aqui, les funcions desenvolupades prèviament:
// desplacament()
// rodaE()
// rodaD()
```

Dins de *setup()* configurem el mode dels pins de sortida i inicialitzem la consola sèrie.

La consola sèrie mostrarà el valor lluminós captat per cada Ldr i ens servirà de mostra per decidir com ajustar el valor de la variable *llindar*.

Dins de *loop()* llegim el valor de cada Ldr i ho mostrem per la consola sèrie.

Posteriorment fem les comparacions i segons sigui, farem avançar el mòbil, girar a la dreta o l'esquerra. Per això comprovem si el nivell de llum és superior al llindar definit i si es més gran que els altres valors. Si es compleix, girarem a dreta, a esquerra o anirem recte, segons correspongui per cada Ldr.

El valor de la variable *llindar* cal canviar-lo segons siguin els valors mesurats per les Ldr's, visibles per la consola sèrie, que depenen de la llum ambiental i la intensitat de la llum a seguir.

Un cop implementat el circuit i el programa, abans del primer ús, cal ajustar els valors de les variables *v* i *llindar*. Amb l'ajust de *v* definim la velocitat per defecte del mòbil, pot valer entre 0 i 100, però, els valors petits, propers a 0 no són efectius ja que els motors no inicien el moviment per valors de tensió petits.

Amb el mòbil connectat amb el cable USB, determinarem el valor adequat de *llindar*. La variable *llindar* indica el nivell de llum, per sobre del qual considerem que és la llum que ha de seguir el mòbil. És un valor que depèn de factors externs, com ara el nivell de llum ambient i el valor de la llum de la llanterna que s'orienta cap al mòbil per que la segueixi.

El programa mostra de forma contínua, a la consola sèrie, els valors llegits per les Ldr's. Observant els valor indicats a la consola sèrie, amb la llum de la llanterna aplicada a les Ldr's o solament amb la llum ambiental, podrem determinar el valor adequat del llindar lluminós a partir del qual han de funcionar els motors. Un cop fet l'ajust de la variable *llindar*, adaptat a l'entorn d'utilització, ja es pot usar el mòbil de forma autònoma desconnectant-lo del cable USB i alimentant-lo solament amb les bateries.

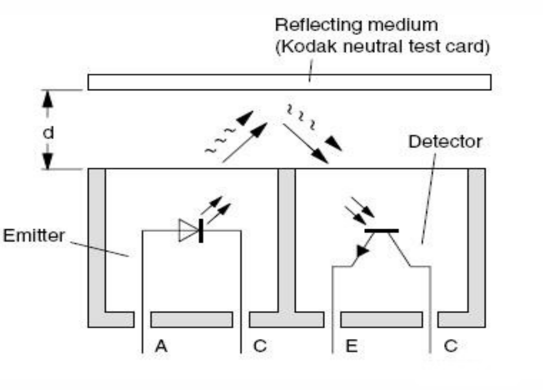
És important que la llum de la llanterna sigui molt més gran que la llum ambiental, per això, el circuit funcionarà millor si la llum ambiental és baixa.

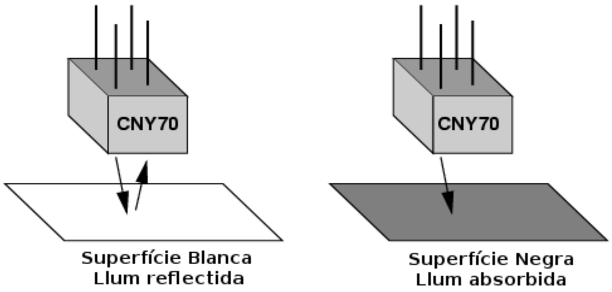
5.14.- Pràctica 14. Mòbil. Seguidor de línies.

Volem implementar el circuit corresponent per fer que el mòbil controlat per Arduino segueixi el recorregut d'una línia negra sobre un fons blanc.

Per això tenim el sensor optoacoblador reflectiu CNY70 que és capaç de determinar si sota d'ell s'hi troba una superfície de color negre o bé una superfície color blanc.

Aquest sensor consta d'un Led emissor de llum IR i d'un fototransistor detector de llum IR. La llum IR del Led emissor es transmet sobre una superfície propera. Si la superfície és de color negre absorbirà tota la radiació lluminosa i no la reflectirà, en canvi, si la superfície és blanca, la llum IR serà reflectida cap al fototransistor. Si sobre el fototransistor hi incideix la llum IR reflectida, aquest entrarà en estat de conducció, de cas contrari es mantindrà en estat de tall.



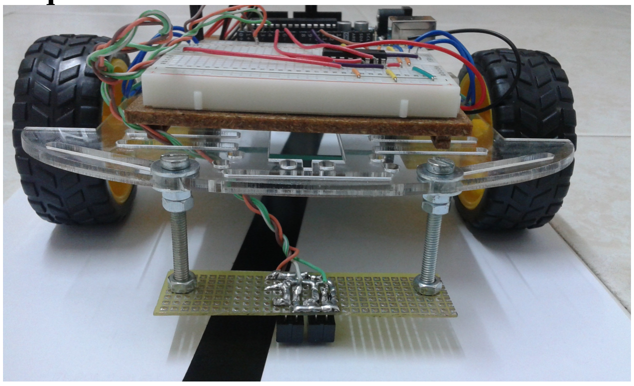


Si la llum es reflectida per la superfície, aquesta incidirà sobre el transistor detector i conduirà entre col·lector i emissor, de cas contrari, no conduirà.

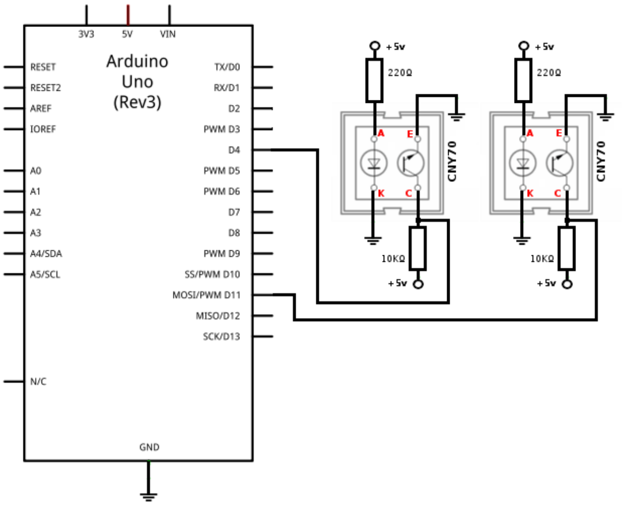
Components: A més dels components de la plataforma mòbil de la pràctica 12, 2 sensors CNY70, 2 resistències de 220Ω i 2 resistències de 10KΩ.

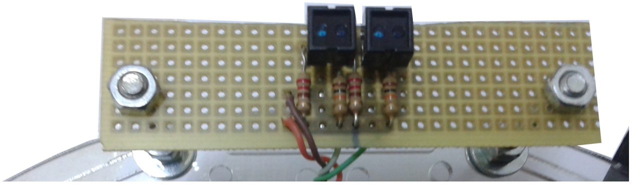
Per construir el mòbil seguidor de línies afegirem al xassís implementat a la pràctica 12 el circuit següent, amb dos sensors CNY70 tal com es veuen connectats a l'esquema.
La ubicació física dels dos sensors ha d'estar a la part davantera d'avanç del mòbil i a una distància molt petita de la superfície on es troba la línia a seguir. Per això cal implementar una placa auxiliar de suport i connexió dels sensors com la que es veu a la imatge següent. Com més petita sigui la distància entre els sensors i la superfície, millor serà la sensibilitat i precisió del mòbil seguidor de línies.

Esquema:



Vista frontal del mòbil amb els dos sensors optoacobladors reflectius ubicats prop de la línia negra sobre fons blanc que ha de seguir.





Vista inferior dels dos sensors CNY70 amb les resistències i connexions connectats sobre una placa auxiliar de prototips fixada al xassís del mòbil.

La separació entre els dos sensors depèn del gruix de la línia negra a seguir. La separació màxima ha de ser com a molt l'ample de la línia, així tindrem la seguretat que en algun moment els dos sensors estaran sobre la línia i el mòbil avançarà recte reduint el moviment lateral. Com més separats estiguin, l'avanç del mòbil produïra un moviment lateral alternat més gran i viceversa.

Programa:

```
#define aRE 10 //pin avanç roda esquerra
#define rRE 9 //pin retrocés roda esquerra
#define aRD 6 //pin avanç roda dreta
#define rRD 5 //pin retrocés roda dreta

#define cny70D 4 //pin sensor reflectiu dreta
#define cny70E 11 //pin sensor reflectiu esquerra

void desplacament(int v=0, char sentit='a', char gir='n', int radi=0);

void setup()
{
 pinMode(aRE, OUTPUT);
 pinMode(rRE, OUTPUT);
 pinMode(aRD, OUTPUT);
 pinMode(rRD, OUTPUT);

 pinMode(cny70D, INPUT);
 pinMode(cny70E, INPUT);
}
```

Definim els pins on es connecten els dos sensors.

Configurem els pins dels sensors com a entrada.

```
void loop() {
  /* Seguim una línia negra sobre fons blanc reflectant.
   * Si el sensor està sobre la zona blanca dona nivell baix.
   * Si el sensor està sobre la zona negra o buit, dona nivell alt.
   * Decidim el moviment segons l'estat dels dos sensors.
   */
   int v=60; //velocitat motors.
  int rflxD=digitalRead(cny70D);
  int rflxE=digitalRead(cny70E);

  if(rflxD==1 && rflxE==1) desplacament(v,'a','n',0); //avanç recte
  else if(rflxD==0 && rflxE==1) desplacament(v,'a','d',50); //gir
esquerra
  else if(rflxD==1 && rflxE==0) desplacament(v,'a','e',50); //gir dreta
  else desplacament(0); //aturat, s'ha sortit de la línia.
}
```

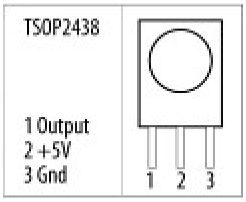
Repetitivament, llegim l'estat dels dos sensors i, segons siguin, avancem, girem o aturem el mòbil.

Si els dos sensors CNY70 estan sobre la línia negra, avancem recte, si els dos estan sobre la zona blanca, s'atura el mòbil. En canvi, si un sensor està sobre la línia negra i l'altre sobre la zona blanca, avancem girant a dreta o esquerra segons convingui per tornar els sensors del mòbil cap a dins de la línia.

El circuit permet altres variants a la que s'ha implementat, només cal canviar el programa adequadament. Per exemple, si volem seguir una línia blanca sobre un fons negre, caldria invertir la lògica de les condicions per decidir el tipus de desplaçament.

5.15.- Pràctica 15. Mòbil. Control amb comandament a distància de TV.

Volem controlar els moviments del mòbil mitjançant un comandament a distància de TV que emet codis amb llum IR. Per això utilitzarem un sensor receptor de llum infraroja model: TSOP2438.



La majoria de comandaments a distancia de TV emeten uns feixos de llum infraroja amb una seqüència d'impulsos de llum de 38kHz[8] (tall-llum), diferent per a cada botó premut del comandament. Cada seqüència detectada pel sensor IR, es tradueix en un valor numèric que pot ser interpretat pel programa de control. Cada fabricant utilitza diferents codis, però, en el nostre cas, els detectarem per als botons que ens siguin necessaris mitjançant la seva lectura en la consola sèrie.

Per utilitzar el sensor IR cal carregar la llibreria: *IRremote.h* que s'encarrega de determinar els codis a partir de les seqüències lluminoses rebudes pel sensor IR segons correspongui pel fabricant del comandament a distància.

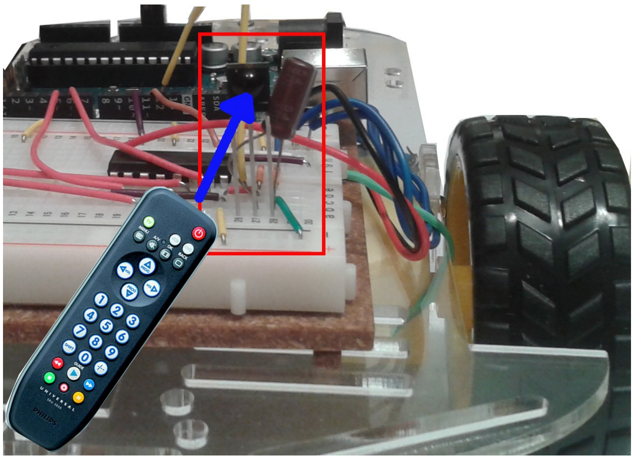
8 Aquesta és una de les freqüències més habituals en molts comandaments de TV, però, també hi ha comandaments que treballen a les freqüències de 30kHz, 36kHz, 40kHz, 56kHz, etc. Si el comandament utilitzat no es detectat pot ser que treballi en una altra freqüència, llavors es pot substituir el sensor per un altre compatible en aquestes freqüències, o bé, canviar de comandament.

En les versions actuals del IDE Arduino, la llibreria *IRremote.h* no funciona adequadament, llavors, cal canviar-la per una altra que si que funcioni. Aquesta altra llibreria la podem descarregar des de la URL: https://github.com/z3t0/Arduino-IRremote, un cop descarregada s'ha de descomprimir i copiar la carpeta obtinguda a dins del directori "libraries" ubicat a la instal·lació del IDE Arduino. També cal esborrar la carpeta RobotIRremote dins el mateix directori.

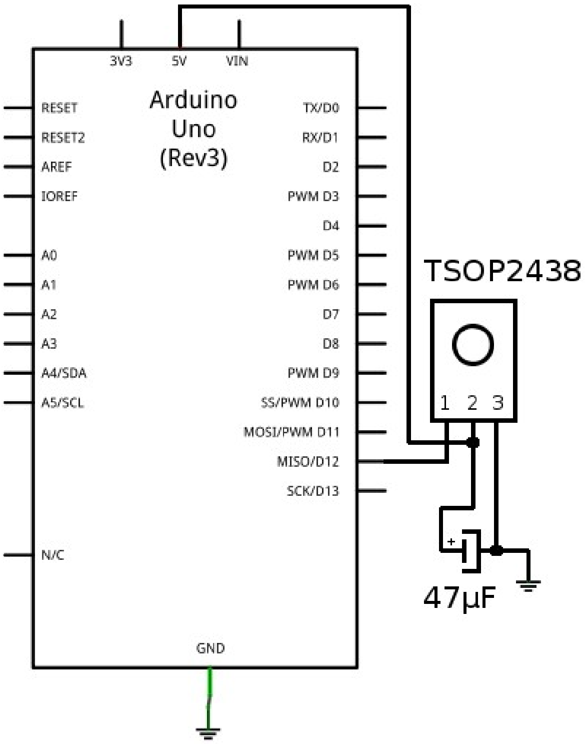
Nota: La llibreria *IRremote.h* utilitza internament el TIMER2, això vol dir que mentre s'usi aquesta llibreria, no es podrà utilitzar la funció PWM en els pins 3 i 11 (però si que es poden usar aquests pins com a entrades-sortides digitals).

Components: A més del components de la plataforma mòbil de la pràctica 12, 1 sensor receptor de IR TSOP2438, 1 condensador de 47µF.

Esquema:

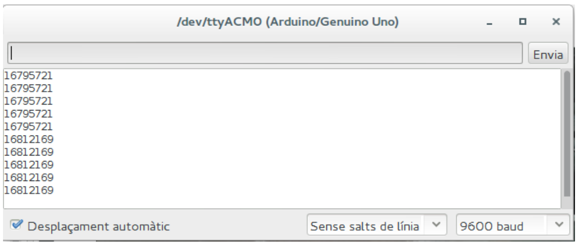


Plataforma mòbil on s'ha afegit, a la part frontal, el receptor IR. El comandament a distància d'un TV s'ha d'encarar cap al sensor IR per que es pugui detectar el codi del botó premut.



Circuit amb el receptor IR, afegit al circuit de la plataforma mòbil de la pràctica 12.

Quan el circuit rebi senyals des del comandament de TV i el programa estigui en funcionament, veurem per la consola sèrie el codi numèric corresponent al botó premut. Aquest codi és el que haurem d'utilitzar per substituir-lo en el programa, segons siguin els botons elegits lliurement, per fer les funcions d'avanç, retrocés, gir dreta, gir esquerra i aturada en el comandament utilitzat.



Programa:

```
#include <IRremote.h>
#define aRE 10  //pin avanç roda esquerra
#define rRE 9  //pin retrocés roda esquerra
#define aRD 6   //pin avanç roda dreta
#define rRD 5   //pin retrocés roda dreta

#define IRpin 12 //pin entrada dades IR

//Codis detectats al comandament TV.
//Els valors de codi són orientatius.
//Cal elegir la tecla per a cada moviment
//i després llegir-ne el codi per la
//consola sèrie. El valor llegit és el que
//s'ha de posar a continuació:
#define AVANCAR 16798291 //tecla avanç.
#define ENRERA 16831187 //tecla enrere.
#define DRETA 16839411 //tecla gir dreta.
#define ESQUERRA 16806515 //tecla gir esq.
#define ATURA 16814739 //tecla atura

void desplacament(int v=0, char sentit='a',
char gir='n', int radi=0);

//definim l'objecte irreceptor indicant
//el pin d'entrada IR, també la variable
//codiRebut, que rebrà les dades llegides.
IRrecv irreceptor(IRpin);
decode_results codiRebut;

void setup() {
  Serial.begin(9600);

  //Habilitem la lectura de dades IR
  irreceptor.enableIRIn();

  pinMode(aRE, OUTPUT);
  pinMode(rRE, OUTPUT);
  pinMode(aRD, OUTPUT);
  pinMode(rRD, OUTPUT);

}

void loop() {
  if(irreceptor.decode(&codiRebut)) {
    Serial.println(codiRebut.value);
    delay(50);

    if(codiRebut.value==AVANCAR)
desplacament(70,'a', 'n', 0);
    else if(codiRebut.value==ENRERA)
desplacament(70,'r', 'n', 0);
```

El programa comença amb la inclusió de la llibreria *IRremote.h* i la definició dels pins usats pel control dels motors i el pin per on es llegeix la senyal del sensor IR. La llibreria *IRremote.h* no és la que porta per defecte Arduino, sinó que és una nova llibreria descarregada d'Internet, tal com s'explica en el paràgraf anterior.

Aquí definim els codis dels botons elegits en el comandament TV per fer les funcions d'avanç, retrocés, gir i aturada. Els valors numèrics són orientatius i els reals s'han de obtenir llegint el valor corresponent a la consola sèrie.

Inicialitzem la recepció IR.

De forma repetitiva, dins de *loop()*, comprovem si s'ha rebut algun codi des del receptor IR, si es així, l'escrivim a la consola. Després comprovem si es un codi reconegut per a fer un moviment mitjançant l'estructura *if..else if*. Si es reconeix

```
    else if(codiRebut.value==DRETA)
desplacament(70,'a', 'd', 0);
    else if(codiRebut.value==ESQUERRA)
desplacament(70,'a', 'e', 0);
    else if(codiRebut.value==ATURA)
desplacament(0);

    irreceptor.resume();
  }
}
//inserir aquí, les funcions desenvolupades prèviament:
// desplacament()
// rodaE()
// rodaD()
```

com a codi per fer un moviment, l'efectuem mitjançant la funció *desplacament()* amb els paràmetres adients de velocitat, gir, direcció, etc.

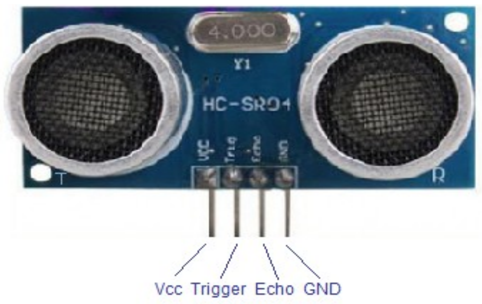
Al final, afegim les funcions *desplacament(), rodaE() i rodaD()* igual com han estat creades a la Pràctica 12.

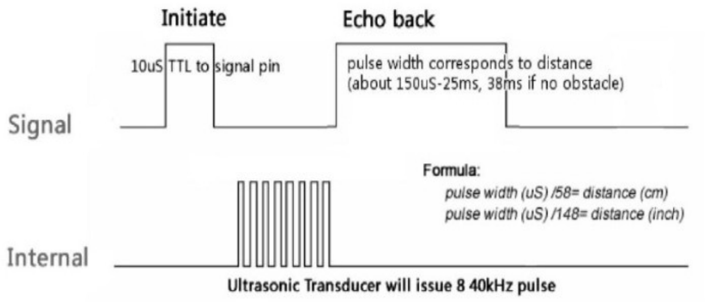
Nota: Alguns comandaments de TV, per mantenir la compatibilitat amb diferents models de receptors TV, emeten diferents codis alternativament per a la mateixa tecla mentre aquesta està premuda. Això no es cap problema pel nostre cas, podem agafar qualsevol dels codis que emet per fer la comparació.

5.16.- Pràctica 16. Mòbil. Detector d'obstacles.

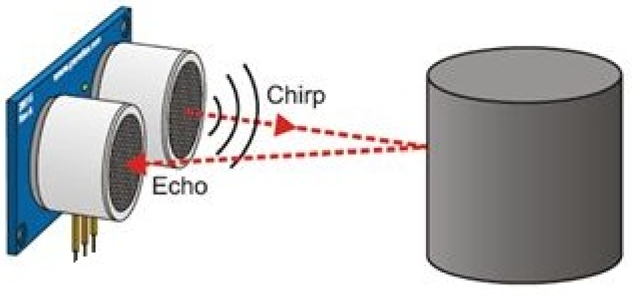
Volem construir un mòbil que avanci contínuament i sigui capaç de detectar si hi ha un obstacle al seu davant. Si detecta un obstacle a una distància menor o igual de 10 cm, sense tocar-lo, el mòbil haurà de tirar enrere, girant, durant 2s i posteriorment, continuar avançant, si l'obstacle ha desaparegut.

Per implementar aquest detector d'obstacles utilitzarem un sensor ultrasònic, concretament el model HC-SR04:





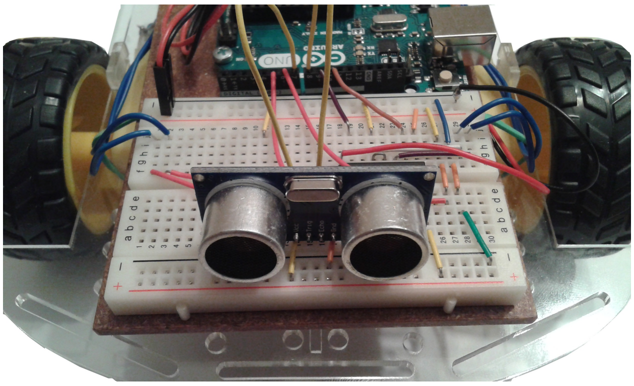
En aplicar un impuls de 10µs (Initiate) al pin Trigger s'inicia l'emissió de 8 polsos d'ultrasò a 40kHz pel transmissor. Un cop emesos els polsos, s'inicia un temps d'espera (Echo back) per rebre el rebot pel receptor, temps que manté el pin Echo a nivell alt mentre no arribi el rebot sonor. El temps d'espera depèn de la distància a l'obstacle.



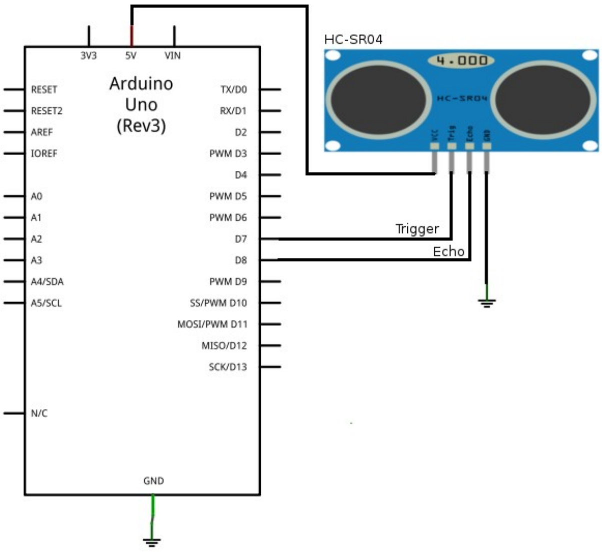
Aquest sensor s'alimenta a 5v (Vcc) i consta d'un transmissor i un receptor de ultrasons (a la freqüència de 40 kHz). Quan s'aplica un impuls pel seu pin Trigger, de 10 µs, emet una ona ultrasònica pel transmissor i espera rebre i detectar el seu rebot pel receptor. El temps que es triga en rebre el seu rebot depèn de la distància que hi hagi fins a l'obstacle. Coneixen la velocitat del so a l'aire (340 m/s), es pot calcular la distància.

Components: A més del components de la plataforma mòbil de la pràctica 12, 1 sensor de distància per ultrasons, HC-SR04.

Esquema:



Imatge de la plataforma mòbil amb el sensor detector d'obstacles ubicat en la part frontal.



Circuit amb el sensor de distància ultrasònic, afegit al circuit de la plataforma mòbil de la pràctica 12.

Per facilitar la lectura de la distància des del programa, implementarem una funció de nom *mesuraDistUltrason()* que, en invocar-la, retorni la distancia de l'obstacle que hi hagi al seu davant, en cm. Per això, la funció envia un pols de 10µs pel pin connectat a Trigger i seguidament es mesura el temps en que el pin Echo està a nivell alt.

Per mesurar aquest temps a nivell alt de Echo, s'utilitza la funció predefinida: ***pulseIn(pin, value)***, on *pin* indica el pin on volem mesurar el temps i *value* indica el nivell (HIGH o LOW) que volem mesurar aquest temps. La funció ens retorna el nombre de microsegons que el *pin* ha estat al nivell indicat.

Programa:

```
#define aRE 10  //pin avanç roda esquerra
#define rRE 9  //pin retrocés roda esquerra
#define aRD 6   //pin avanç roda dreta
#define rRD 5   //pin retrocés roda dreta

#define TRIG 7  //Pin Trig del sensor
ultrasonic
#define ECHO 8  //Pin Echo del sensor
ultrasonic

void setup() {
  pinMode(aRE, OUTPUT);
  pinMode(rRE, OUTPUT);
  pinMode(aRD, OUTPUT);
  pinMode(rRD, OUTPUT);
```

Definim noms constants pels pins on estan connectats Trigger i Echo del sensor ultrasònic.

Dins de *setup()* inicialitzem el mode de treball dels pins.
Inicialitzem com a sortida i com a entrada els pins Trigger i Echo, respectivament.

```
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);

  Serial.begin(9600);
}

void loop() {
  //distCm conté la distancia fins
l'obstacle en cm.
  int distCm=mesuraDistUltrason(TRIG,
ECHO);

  Serial.print(distCm);
  Serial.println(" cm.");

  int v=60;
  if(distCm<10) {
    desplacament(v, 'r', 'e', 50);
    delay(2000);
  }
  else desplacament(v, 'a', 'n', 0);
}

int mesuraDistUltrason(int tr, int ec) {
  //Retorna la distancia a obstacle en cm.
  //tr: pin trigger, ec: pin echo
  digitalWrite(tr, LOW);
  delayMicroseconds(2);
  digitalWrite(tr, HIGH);
  delayMicroseconds(10);
  digitalWrite(tr, LOW);

  return pulseIn(ec,HIGH)/58;
}

//inserir aquí, les funcions desenvolupades
prèviament:
// desplacament()
// rodaE()
// rodaD()
```

Dins de *loop()* determinem la distància del possible obstacle que es pugui trobar davant el sensor invocant la funció *mesuraDistUltrason()* indicat el numero dels pins on s'ha connectat el sensor. La mesura efectuada la mostrem per la consola sèrie.
El mòbil estarà constantment en desplaçament recte endavant, excepte quan té un obstacle a una distància inferior a 10 cm. En aquest cas, fa un moviment enrere girant durant 2 s. i torna a efectuar el moviment endavant sempre que ja no hi hagi obstacle.

La funció *mesuraDistUltrason()* fa una mesura de distància i la retorna en cm. Per això, envia un impuls de 10µs a Trigger per iniciar l'emissió ultrasònica i espera el rebot. Mentre no arriba el rebot, el pin Echo estarà a nivell alt. Llavors, comptarem, amb pulseIn(), el temps (en µs) en que es troba el pin Echo a nivell alt. El temps mesurat el dividim per 58, tal com s'indica a les característiques del sensor per tenir el resultat en cm.
Al final, afegim les funcions *desplacament(), rodaE() i rodaD()* igual com han estat creades a la Pràctica 12.

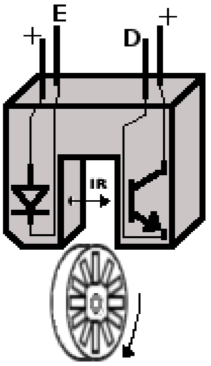
5.17.- Pràctica 17. Mòbil. Control rotacions motor amb barrera de llum.

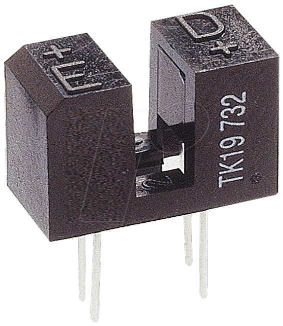
Volem monitoritzar i controlar la velocitat de desplaçament del mòbil i el nombre de voltes fetes per les seves rodes.

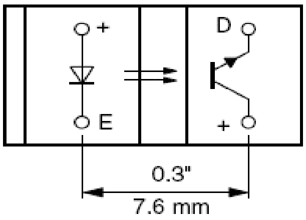
Mitjançant dos barreres de llum (sensor òptic transmissiu TCST1103 de Vishay) implementarem un encoder rotatiu senzill a cada roda per poder comptar les voltes fetes. Un encoder rotatiu es un dispositiu sensor de moviment angular que detecta l'angle girat, les voltes fetes i el sentit del gir. En la nostra senzilla implementació, no detectarem el sentit de gir.

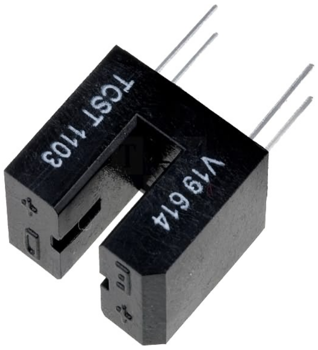
Amb aquest sensor, podem implementar un sistema realimentat del mecanisme de tracció de la plataforma mòbil, de forma que podem obtenir el valor del resultat de sortida del sistema com a resposta a la tensió aplicada als motors.

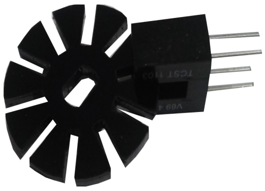
El sensor TCST1103 consisteix en un Led emissor de llum IR que incideix sobre un fototransistor, sensible a la mateixa llum IR, que es troba situat a una petita distància d'uns 7'6 mm. Entre aquesta distància de separació hi fem girar una roda perforada acoblada a l'eix de la roda motriu, de tal manera que en girar, el raig de llum IR podrà passar pels forats de la roda perforada, o no. Si el llum pot passar, activarà el fototransistor i en el col·lector obtindrem el conjunt d'impulsos elèctrics (tall - no tall, del raig IR) corresponents al gir de la roda perforada. Coneixen el nombre de forats a la roda perforada, el radi de la roda motriu i el temps, podem comptar els impulsos. Així calcularem l'**angle girat**, el **nombre de voltes**, la **velocitat de gir per roda** i **la velocitat de desplaçament del mòbil.**





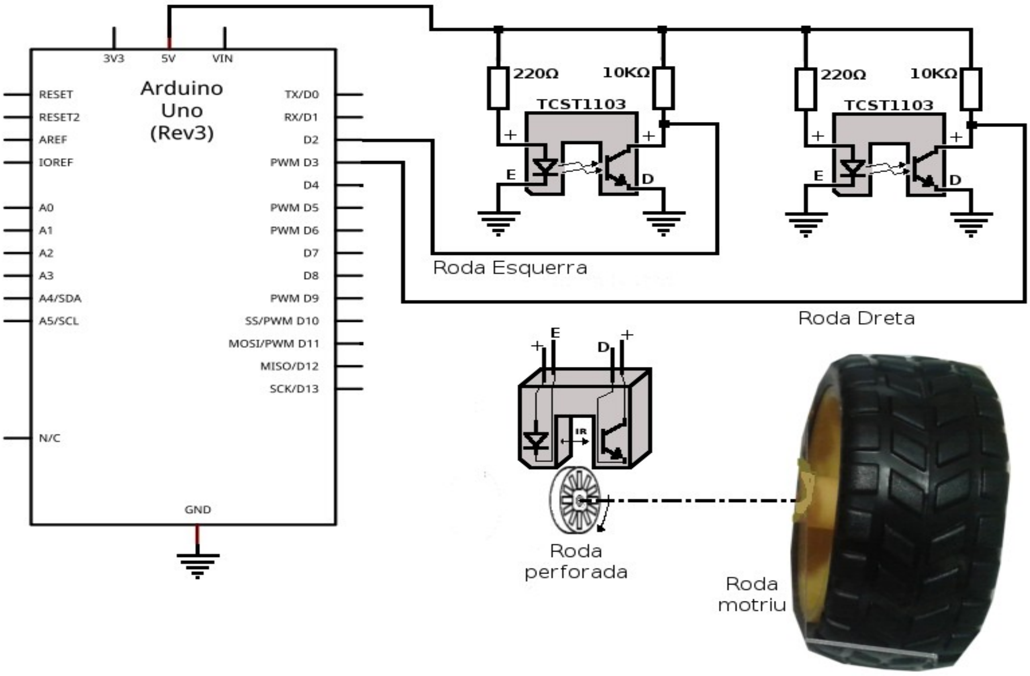


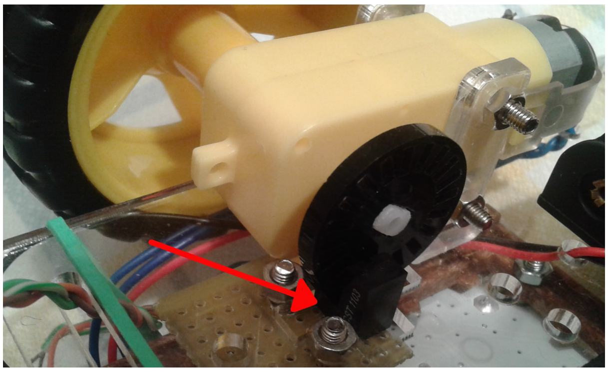


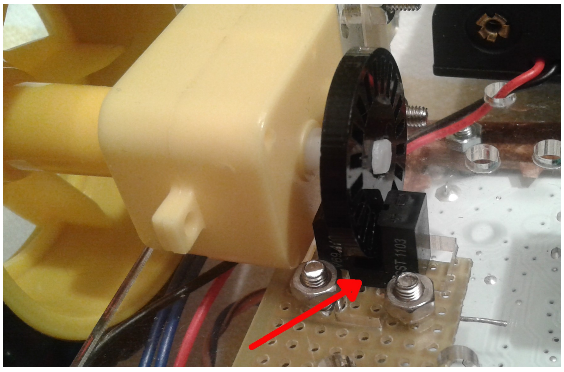


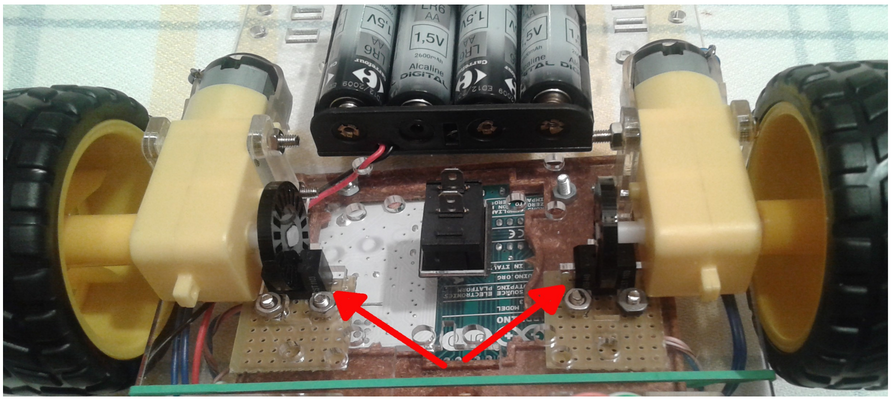
Components: A més del components de la plataforma mòbil de la pràctica 12, 2 sensors barrera de llum TCST1103, 2 resistors de 10KΩ i 2 resistors de 220Ω.

Esquema:









Vista inferior de la plataforma mòbil amb els detalls dels sensors òptics transmissius que formen les barreres de llum i les rodes perforades que giren a la mateixa velocitat de les rodes motrius conformant els encoders òptics rotatius.

En els programes fets fins ara en pràctiques anteriors es passen tot el temps repetint milers de vegades les instruccions que es troba dins de *loop()*, d'aquesta forma es va consultant, contínuament i seqüèncialment, el valor dels sensors i s'actua segons correspongui. Aquest mètode és conegut com a **Polling**, ja que el programa consulta contínuament els sensors. Aquest mètode és vàlid en la majoria d'ocasions en que no es requereix immediatesa en la lectura del sensor per que l'scan del bucle ja es prou ràpid, tot i tenir variacions de temps importants segons les tasques que es faci.

La seqüència de generació dels impulsos en els fototransistors és molt elevada quan es desplaça a alta velocitat. La seva lectura i compteig dels impulsos dins de *loop()* pel mètode Polling no és eficient, per que es requereix atenció constant per no perdre cap impuls produït, sinó, els càlculs de voltes i velocitat serien imprecisos.

És necessari disposar d'un sistema de lectura immediat dels impulsos, que sigui independent de l'activitat que es faci dins de *loop()* i amb això garantir la precisió de les lectures i els càlculs.
Per aconseguir això utilitzarem el mètode **Interrupt** (complementari al mètode Polling), disponible en els pins 2 i 3 d'Arduino Uno. Aquest mètode consisteix en que l'existència d'un canvi en algun d'aquests pins, provoqui una Interrupció de l'activitat normal del programa per atendre el que passa en el pin. Aquesta atenció s'efectuarà mitjançant una funció definida expressament per atendre la interrupció mitjançant la definició feta a *attachInterrupt()*.
En el programa següent utilitzem dos funcions: *contImpE()* i *contImpD()*, una per a cada encoder rotatiu que atendran la interrupció originada en el pins 2 i 3, és a dir, cada impuls originat en cada encoder.

Programa:

```
#define aRE 10  //pin avanç roda esquerra
#define rRE 9  //pin retrocés roda esquerra
#define aRD 6  //pin avanç roda dreta
#define rRD 5  //pin retrocés roda dreta

#define sotrE 2  //pin sensor òptic transmissiu rotació esquerra
#define sotrD 3  //pin sensor òptic transmissiu rotació dreta

void desplacament(int v=0, char sentit='a', char gir='n', int radi=0);


//definició global de variables
//les variables modificables dins les funcions
//d'atenció a les interrupcions també es declaren
// volatile
int nforatsDisc=20;
int radiRoda=33; //en mm.
volatile int contE=0;
volatile int contD=0;
volatile float voltesE=0.0;
volatile float voltesD=0.0;


void setup() {
  pinMode(aRE, OUTPUT);
  pinMode(rRE, OUTPUT);
  pinMode(aRD, OUTPUT);
```

Definim els pins on connectem la sortida d'impulsos de cada sensor encoder.

Les variables globals seran usades en les funcions d'atenció a les interrupcions i a la funció *loop()*. Si les variables són modificables dins les funcions d'atenció a les interrupcions, també les declarem com a *volatile*.

Configurem els pins on s'han connectats els sensors encoders rotatius com a entrada.

```
  pinMode(rRD, OUTPUT);

  pinMode(sotrE, INPUT);
  pinMode(sotrD, INPUT);

  Serial.begin(9600);

  attachInterrupt(digitalPinToInterrupt(sotrE),
contImpE, RISING);
  attachInterrupt(digitalPinToInterrupt(sotrD),
contImpD, RISING);

}
```

Amb *attachInterrupt()* configurem que les entrades dels pins dels dos sensors encoders rotatius siguin ateses per les funcions (*contImpE()* i *contImpD()*) quan es produeixi la interrupció. La interrupció es produirà quan existeixi un flanc ascendent al pin (*Rising*)

```
void loop() {
  float voltesEant=0.0;
  float voltesDant=0.0;
  float velE=0.0;
  float velD=0.0;
  float velMitja=0.0;
  long millisAnt=0;
  int t=3; //marge de temps en s. per calcular la vel.

  //A partir dels impulsos i voltes comptats,
  //calculem velocitats:
  if(millis()>(millisAnt+t*1000)) {
    float sec=(float)(millis()-millisAnt)/1000;
    millisAnt=millis();
    velE=(float)2*PI*radiRoda*(voltesE-voltesEant)/sec;
    velD=(float)2*PI*radiRoda*(voltesD-voltesDant)/sec;
    velMitja=(velE+velD)/2;
  }
  voltesEant=voltesE;
  voltesDant=voltesD;
```

Calculem les velocitats de cada roda i mitjana cada t segons.

```
   Serial.print("Roda Esquerra. Impuls.:");
   Serial.print(contE);
   Serial.print(" Voltes:");
   Serial.print(voltesE);
   Serial.print(" Vel.:");
   Serial.print(velE);
   Serial.print("mm/s ");
   Serial.print(" | Roda Dreta. Impuls.:");
   Serial.print(contD);
   Serial.print(" Voltes:");
   Serial.print(voltesD);
   Serial.print(" Vel.:");
   Serial.print(velD);
   Serial.print("mm/s");
   Serial.print(" | Vel.Mitja:");
   Serial.print(velMitja);
   Serial.println("mm/s");

   desplacament(100);
}
```

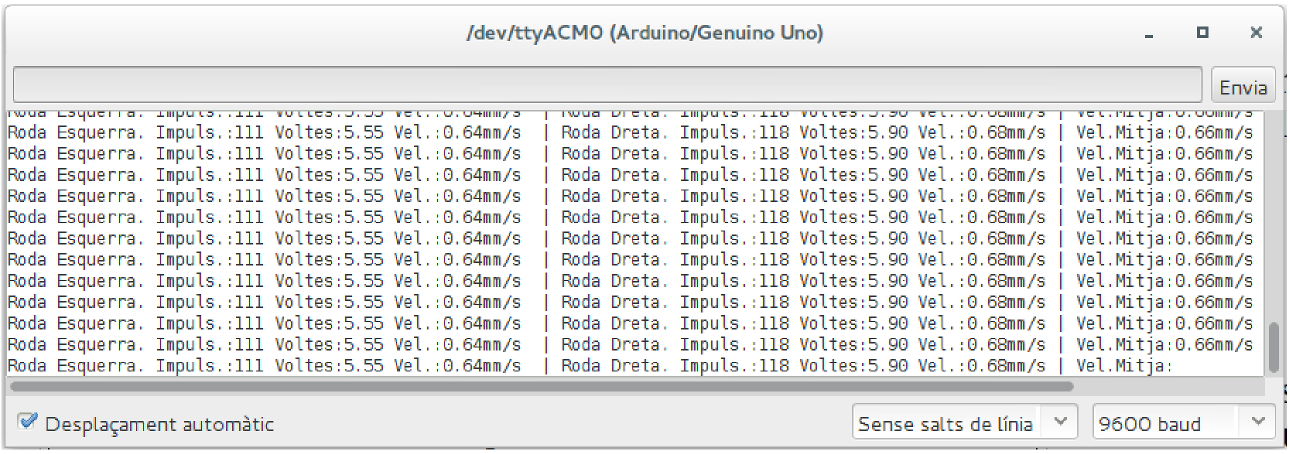
Un cop fets els càlculs de velocitat, els mostrem per la consola sèrie.

```
void contImpE() {
  ++contE;
  voltesE=(float)contE/nforatsDisc;
}
```

Implementació de les funcions *contImpE()* i *contImpD()* que atendran a les interrupcions

```
void contImpD() {
  ++contD;
  voltesD=(float)contD/nforatsDisc;
}

//inserir aquí, les funcions desenvolupades prèviament:
// desplaçament()
// rodaE()
// rodaD()
```

produïdes. Cada funció, per a cada roda, incrementa el valor del comptador de impulsos i calcula en nombre de voltes fetes per la roda a partir del nombre de forats del disc perforat.



L'exemple implementat a la pràctica mostra per la consola sèrie, el valor del nombre de impulsos comptats, el nombre de voltes fet i la velocitat lineal per a cada roda i la velocitat mitjana de totes dues rodes.

Aquesta és una mínima implementació, útil per mostrar com funciona la utilització dels encoders de gir i velocitat angular.

L'exemple bàsic que s'ha desenvolupat ha de servir per poder implementar exemples pràctics més elaborats com les següents:

- Fer avançar la plataforma mòbil un determinat nombre de voltes de roda.
- Fer avançar la plataforma mòbil a una velocitat lineal determinada.
- Fer girar les rodes un determinat angle comptant els impulsos.
- etc.

5.18.- Pràctica 18. Mòbil. Comunicar Arduino per Bluetooth. Control des de ordinador PC i Smartphone Android.

La comunicació Bluetooth és un sistema de intercomunicació de dades entre dispositius propers, vinculats entre ells, de forma inalambrica, a la freqüència de 2'4GHz.

En aquesta pràctica és pretén controlar els moviments de la plataforma mòbil, implementada a la pràctica 12, mitjançant un mòdul d'intercomunicació Bluetooth, models HC-05 o HC-06.

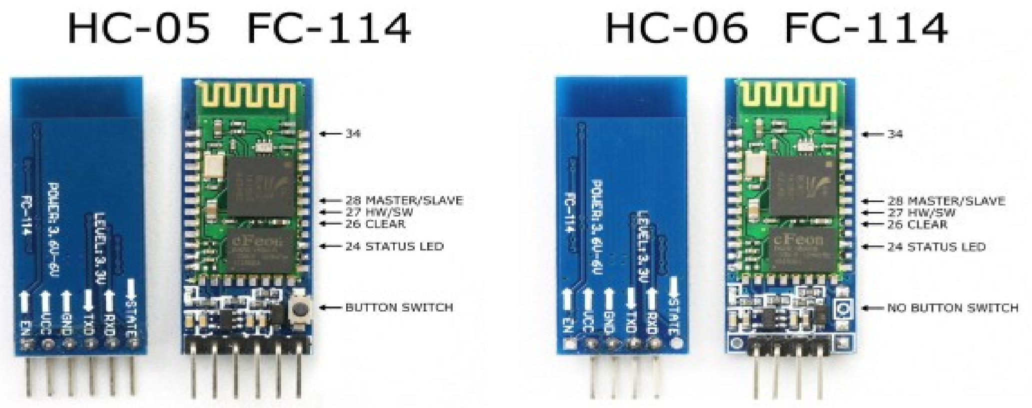
Aquests mòduls faciliten la connexió de la transmissió de dades Sèrie, RXD i TXD, via Bluetooth. Per tant, inicialment farem un programa que permetrà el control del mòbil per la connexió Sèrie, tal com ja havíem fet en alguna pràctica anterior. També farem les proves del correcte funcionament del programa controlant la plataforma mòbil per la connexió sèrie. Quan tot estigui provat, llavors connectarem el mòdul Bluetooth que substituirà al cable USB en l'establiment de la connexió sèrie.

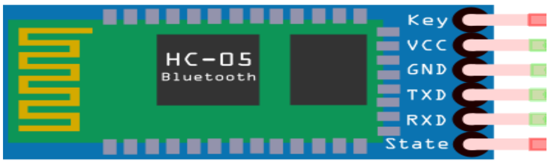
Com que tant el mòdul Bluetooth com el cable USB, utilitzen la mateixa connexió sèrie d'Arduino, no podrem tenir connectats tots dos sistemes simultàniament ja que s'interferirien entre ells. Per això el mòdul Bluetooth tindrà desconnectats els pins corresponents a TXD i RXD mentre es faci la transferència del programa a Arduino i les proves i depuració del programa amb la consola sèrie.

Un cop s'hagi carregat i comprovat el correcte funcionament del programa, aturarem la consola sèrie, desconnectarem el cable USB i alimentarem l'Arduino amb una bateria autònoma. Llavors, ja podrem connectar els pins TXD i RXD del mòdul Bluetooth. A partir d'aquesta connexió, el control ja es pot fer remotament mitjançant la vinculació del Bluetooth de l'Arduino de la plataforma mòbil a un altre dispositiu Bluetooth.

També veurem com vinculem la plataforma mòbil amb Bluetooth a un ordinador i a un smartphone Android. En cada cas serà possible fer el control de la plataforma mòbil enviant comandaments textuals per fer-la avançar, retrocedir, girar a dreta i esquerra i aturar-lo.

Els mòduls Bluetooth HC-05 i HC-06 són dispositius compostos que transmeten al dispositiu on s'ha vinculat, mitjançant l'antena que porten integrada, les dades rebudes i enviades pels pins de connexió sèrie TXD i RXD de Arduino.



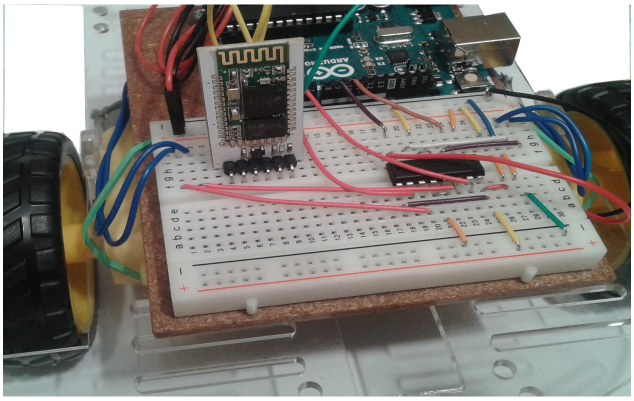


Els dos models són gairebé iguals, però, el HC-05 té dos pins de connexió addicionals (Key i State) afegits als dos extrems, que en aquesta pràctica no utilitzarem, a més el model HC-05 es pot configurar com a master o slave i el HC-06 com a slave.

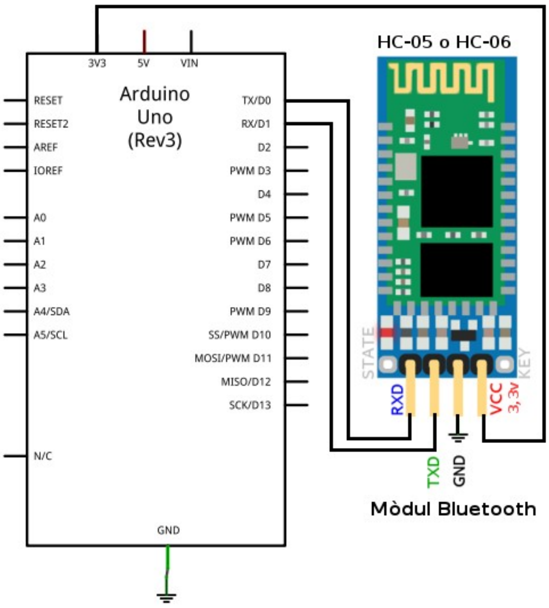
En tots dos models tenim els pins anomenats Vcc, Gnd, TXD i RXD ubicats en la mateixa posició, aquests són els que connectarem al nostre circuit. La tensió aplicada a Vcc és de 3'3v.
Com que en la pràctica implementada utilitzem el mòdul en el seu mode slave, ens serà indiferent utilitzar-ne un model o altre.

Components: A més del components de la plataforma mòbil de la pràctica 12, un mòdul Bluetooth tipus: HC-05 o HC-06.

Esquema:



Imatge de la plataforma mòbil amb l'afegit del mòdul bluetooth.



Nota: TXD i RXD del mòdul Bluetooth han d'estar desconnectats mentre es fa la càrrega del programa pel cable USB i les proves de funcionament del programa amb la consola Sèrie. Un cop carregat el programa i fetes les proves, ja es pot connectar TXD i RXD, havent desconnectat prèviament el cable USB (si més no, no utilitzant la consola sèrie).

Programa:

```
#define aRE 10  //pin avanç roda esquerra
#define rRE 9  //pin retrocés roda esquerra
#define aRD 6   //pin avanç roda dreta
#define rRD 5   //pin retrocés roda dreta

void desplacament(int v=0, char sentit='a', char
gir='n', int radi=0);
```

El programa comença amb la definició dels pins dels motors i la declaració de la funció *desplacament()* i els seus paràmetres per defecte.

```
void setup() {
  pinMode(aRE, OUTPUT);
  pinMode(rRE, OUTPUT);
  pinMode(aRD, OUTPUT);
  pinMode(rRD, OUTPUT);
  pinMode(13, OUTPUT);
  Serial.begin(9600);

  //Procés configuració inicial, per HC06 o HC05:
  digitalWrite(13, HIGH);
  Serial.print("AT");
```

Inicialitzem la comunicació sèrie.

```
  delay(1000);
  //nom bluetooth:
  Serial.print("AT+NAME"); Serial.print("PRBT");
  delay(1000);
  //contrasenya:
  Serial.print("AT+PIN"); Serial.print("1234");
  delay(1000);
  //bauds 9600:
  Serial.print("AT+BAUD"); Serial.print('4');
  delay(1000);
  digitalWrite(13, LOW);
}

void loop() {
  int v=60;
  String entCad;

  if(Serial.available()) {
    while(Serial.available()) {
      char c = (char)Serial.read(); //llegim un char
      entCad += c; //fem un String
    }
    Serial.println(entCad); //eco consola

    if(entCad == "a") { //avanç
        desplacament(v);
    } else if(entCad == "r"){ //retrocés
        desplacament(v, 'r');
    } else if(entCad == "d"){ //gir dreta
        desplacament(v, 'a', 'd', 101);
    } else if(entCad == "e"){ //gir esquerra
        desplacament(v, 'a', 'e', 101);
    } else if(entCad == "s"){ //stop
        desplacament();
    }
    entCad = "";
  }
}

//inserir aquí, les funcions desenvolupades
prèviament:
// desplacament()
// rodaE()
// rodaD()
```

Inicialitzem, mitjançant l'enviament de comandes AT, algunes dades de configuració inicial del mòdul Bluetooth. Configurem el nom, la contrasenya i la velocitat de transmissió.
Mentre s'està fent la configuració inicial, el Led integrat de la placa (pin 13) estarà il·luminat.

Dins de loop() estarem contínuament comprovant si s'ha rebut alguna comanda de control del mòbil mitjançant la comunicació sèrie.

En cas de rebre una comanda, es compara si correspon a una de les programades i si es així, es fa l'acció corresponent. Les comandes vàlides són:

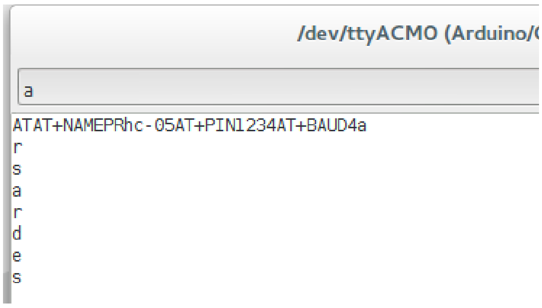
a – Inicia l'avanç
r – Inicia el retrocés.
d – Inicia el gir a la dreta sobre el seu eix.
e – Inicia el gir a l'esquerra sobre el seu eix.
s – Atura qualsevol moviment.

Un cop implementat el circuit addicional Bluetooth, al que ja tenim de l'estructura de la plataforma mòbil i, havent afegit el programa indicat prèviament, podem començar a fer les proves de funcionament.

Inicialment, mantindrem les connexions TXD i RXD del mòdul BT desconnectades per evitar les interferències amb la connexió sèrie d'Arduino i l'ordinador per carregar i depurar el programa. Per tant, treballarem amb el mòdul BT desconnectat i simularem el seu funcionament per la consola sèrie del IDE.

En iniciar el programa, per la consola sèrie podrem veure les comandes AT que el programa envia al mòdul BT, però, en aquest cas no són efectives, ja que el mòdul està desactivat.
Posteriorment, podrem enviar les comandes de control del moviment i verificar que efectivament, la plataforma mòbil fa els moviments de desplaçament esperats.

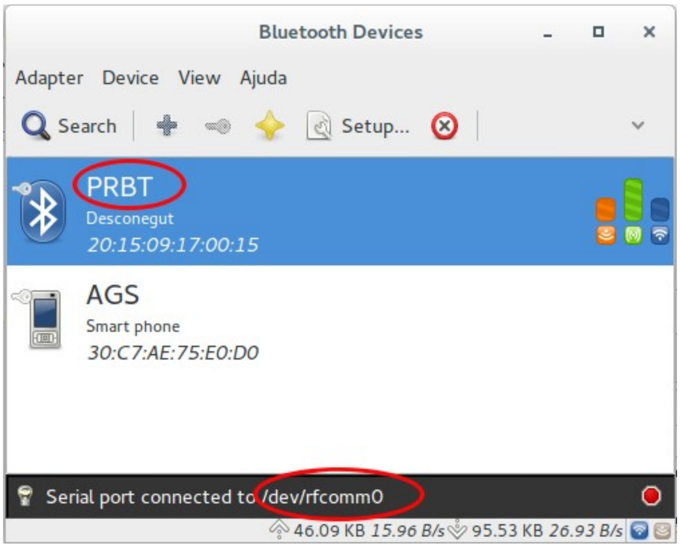
La imatge següent és la vista de la consola sèrie, fent les proves prèvies, on es veu els comandaments AT que s'enviarà al mòdul BT a l'inici i l'enviament manual de comandes *a*, *r*, *d*, *e* i *s*, per fer els moviments de la plataforma mòbil. Per cada comanda, el mòbil s'ha de moure segons s'espera, encara que en un petit recorregut, ja que està limitat pel cable de la connexió USB que facilita la comunicació sèrie.



Un cop provat el programa manualment des de la consola sèrie, substituirem la connexió sèrie pel mòdul Bluetooth. Ara la connexió sèrie s'estableix pel mòdul BT i cal desconnectar el cable USB per evitar interferències. La plataforma mòbil ja tindrà autonomia de moviment ja que l'alimentarem amb bateries.

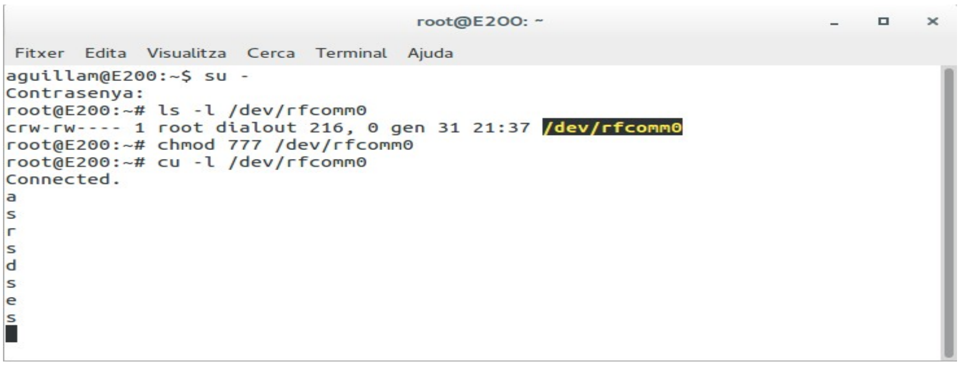
Els passos següents consisteixen en vincular la plataforma mòbil mitjançant Bluetooth amb un ordinador amb el SO Linux i un smartphone Android i controlar els seus moviments a distància.

Vinculació amb un ordinador:
Començarem per l'exemple d'un ordinador amb S.O. Linux, en farem la vinculació i el control des de la consola de comandaments amb un programa de comunicació sèrie. Des de l'ordinador accedim a la configuració i vinculació de dispositius Bluetooth.



En el gestor de dispositius Bluetooth de l'ordinador veiem com el dispositiu apareix amb el nom PRBT, ens vinculem amb ell com a dispositiu Sèrie introduint la contrasenya de vinculació i, un cop enllaçat, ens mostra que el port sèrie que li ha assignat en el nostre ordinador d'exemple és: /dev/rfcomm0 (en un altre ordinador pot ser diferent).

Ara, des del terminal de comandaments de l'ordinador podem obrir un programa de comunicació sèrie, per exemple el 'cu', i li indiquem el port per on comunicar-se, en aquest cas el: /dev/rfcomm0 (prèviament, si es convenient, cal tenir privilegis de root i donar permisos 777 al dispositiu sèrie).



Altres alternatives a: **cu -l /dev/rfcomm0** , poden ser:

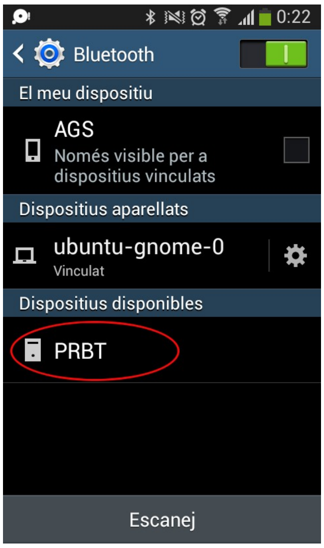
screen /dev/rfcomm0

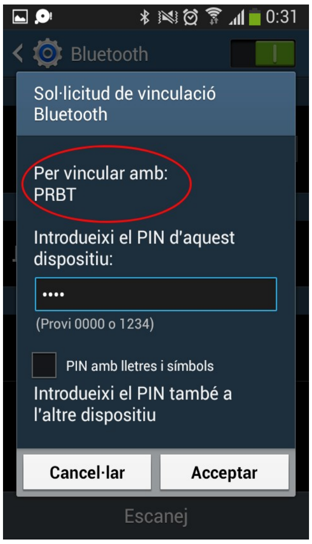
minicom

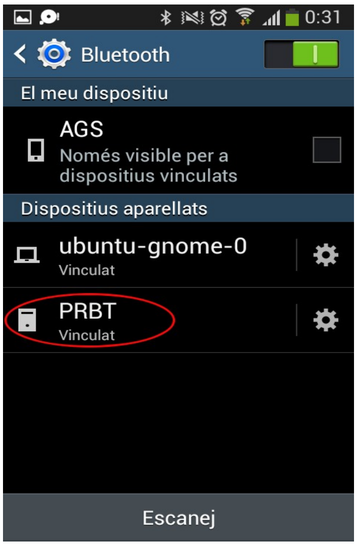
aquests programes de comunicació sèrie de l'ordinador es poden instal·lar des dels repositoris habituals de la distribució Linux utilitzada, en cas que no estiguin ja instal·lats.

Vinculació amb un Smartphone Android:

També podem vincular el nostre dispositiu BT amb un Smartphone tal com es veu en les imatges següents. Però, amb aquesta vinculació, no és suficient per controlar la plataforma mòbil via BT. Fa falta un programa específic que, mitjançant aquesta connexió, pugui enviar els comandaments de control al mecanisme mòbil per fer-ne els moviments.

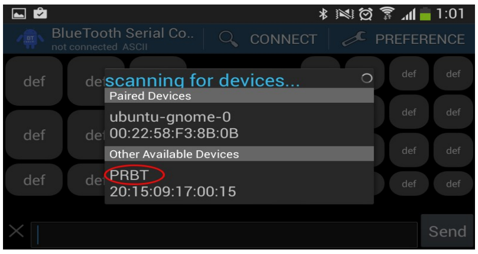


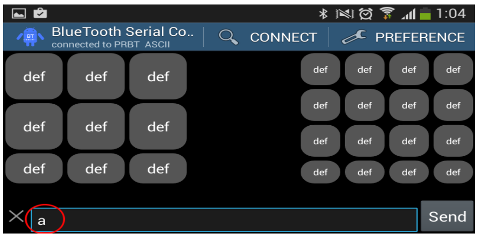


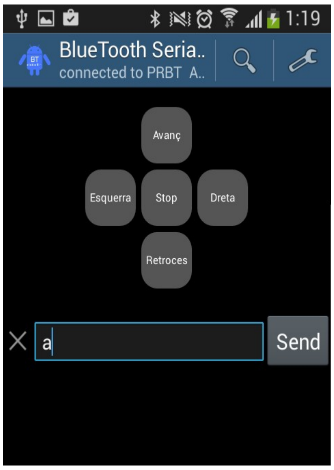


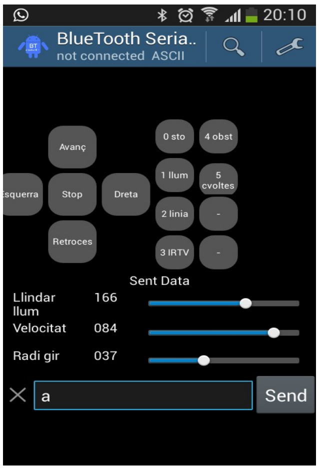
Existeixen al Play Store d'Android algunes App's que han estat creades per facilitar l'enviament d'ordres de control sèrie via BT. Es poden trobar facilment fent una cerca amb paraules com "control bluetooth arduino". Una d'aquestes App's és: **BlueTooth Serial Controller**, elegida, d'entre les moltes disponibles, per fer-ne ús en les captures següents.

Aquesta App, i altres similars, disponible al Play Store d'Android, esta dissenyada per enviar comandaments via Bluetooth a dispositius controlats per Arduino, Picaxe i altres dispositius microcontroladors. Té una interfície que és configurable i es pot ajustar per definir exclusivament els botons necessaris per enviar les instruccions corresponents al dispositiu a controlar via sèrie per mitjà de Bluetooth. Les imatges següents mostren captures de la App indicada fent la vinculació amb el dispositiu, l'enviament de comandaments i la personalització de botons i ordres per la nostra pràctica:









La App disposa de múltiples opcions de configuració personalitzada, de tal manera que podem mostrar/ocultar els botons que vulguem i configurar cadascun d'ells per que, en preme'l, faci l'enviament del comandament de control de moviment a la plataforma mòbil.

Ampliacions:

1.- El mòdul Bluetooth HC-05 pot treballar com a màster o esclau (a la pràctica s'utilitza en el mode per defecte, com esclau). Això vol dir que, a més de connexions amb PC's i Smartphones, també podem fer connexions punt a punt entre dos HC-05, un com a màster i l'altre com esclau, cadascun d'ells controlats per una placa Arduino o similar. Així és poden comunicar dos Arduinos entre si via Bluetooth.

2.- L'exemple d'implementació del control des de l'ordinador PC i Smartphone Android s'ha efectuat en la seva més mínima expressió, mitjançant terminals enllaçats amb la connexió Bluetooth efectuada. S'ha fet així per no complicar l'enteniment de la part fonamental de la pràctica.

Per això, els exemples de control no són molt vistosos. En aplicacions reals més complexes, el més habitual és que el control de la connexió es faci mitjançant programes implementats específicament per a PC o Smartphone, millorant la interfície de comunicació i control.

La programació d'aquestes aplicacions per a PC i Smartphone està en un àmbit de coneixements diferent del que s'estudia en aquest dossier, per tant, no se'n fa cap exemple, però, no comporta més complexitat que la de la programació de la interfície sèrie implementada amb Bluetooth mitjançant un llenguatge d'alt nivell tipus C, C++, Java, Python, etc. adient per al PC, l'Smartphone o qualsevol altre dispositiu amb Bluetooth.

5.19.- Pràctica 19. Sensor de so.

Volem que un led s'il·lumini quan es detecta un so brusc i intens, si està prèviament apagat, o bé, s'apagui en les mateixes condicions si està prèviament il·luminat.

Per això utilitzarem un sensor de so. El sensor de so és un dispositiu format per un micròfon i un pre-amplificador d'àudio o BF. També disposa d'un potenciòmetre ajustable per prefixar el llindar de comparació o el guany. Generalment s'alimenten a la tensió de 5v.

Podem tenir dos tipus de sortida: sortida digital i sortida analògica. Si es de sortida digital ens proporciona un nivell lògic '1' o '0' segons si el so captat és superior o no al valor del llindar de referència indicat pel potenciòmetre ajustable. Si és de sortida analògica ens proporciona un senyal analògic corresponent al so captat prèviament amplificat al guany ajustat amb el potenciòmetre ajustable. Hi han sensors de so que disposen de les dos sortides: la analògica i la digital, per exemple, el model: KY-037.

En el cas de sortida analògica, connectarem el senyal a un pin d'entrada analògic (A0 ... A5) i en el programa haurem d'establir el nivell de llindar del so per comparar-ho amb el senyal d'àudio.

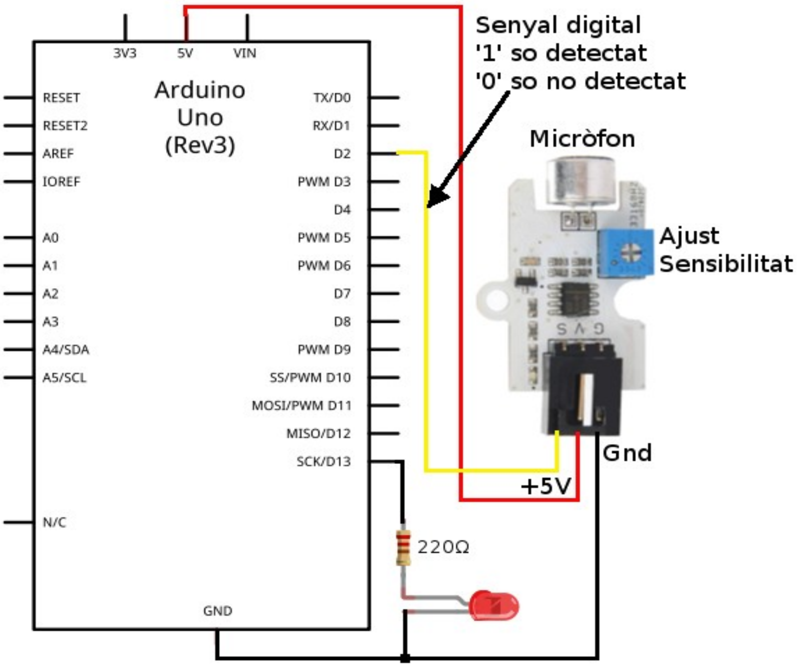
En el cas de sortida digital, connectarem el senyal a un pin d'entrada digital (D0 ... D13) i, segons sigui el valor llegit sabrem si s'ha superat la referència del nivell de so.

En aquesta pràctica utilitzem un sensor de so de sortida digital de la sèrie Octopus. Per tant, connectarem el senyal a una entrada digital i amb el potenciòmetre ajustable del sensor ajustem el nivell de llindar de commutació.

Crearem el programa de control utilitzarem el mètode *Interrupt* per detectar el moment en que es produeix el so. És a dir, enlloc d'estar consultant constantment l'estat del sensor des de la funció *loop()* (mètode *Polling*), el que farem és un programa que atengui a una interrupció originada en el pin 2. La interrupció s'originarà en el moment que hi hagi un canvi en el nivell del senyal provinent del sensor de so. Quan això passi, es detectarà i s'invocarà una funció per atendre aquesta interrupció que s'encarregarà de canviar l'estat del Led connectat al pin 13.

Components: 1 sensor de so amb sortida digital de la sèrie octopus, 1 resistència de 220Ω, 1 Led.

Esquema:



A l'esquema, connectem el senyal del sensor de so al pin 2 d'Arduino ja que, aquest pin, al igual que el pin 3, és pot programar per atendre interrupcions externes.

Programa:

```
#define sSo 2
#define Led 13

volatile bool actv=false;

void setup() {
 pinMode(Led,OUTPUT);
 pinMode(sSo, INPUT);
 attachInterrupt(digitalPinToInterrupt(sSo), detSo, CHANGE);
 Serial.begin(9600);
 digitalWrite(Led, LOW);
}

void loop() {
 Serial.println(actv);
}

void detSo() {
 actv=!actv;
 digitalWrite(Led, actv);
}
```

Comencem definint els pins on connectem el senyal del sensor de so (*sSo*) i el Led.

Declarem la variable *actv* que contindra l'estat que canvia cada cop que s'activa el sensor.

Dins de *setup()* definim el mode dels pins, inicialitzem la consola sèrie i apaguem el Led.
Amb *attachInterrupt()*, també establim que el pin 2 generarà una interrupció cada cop que canviï el seu nivell invocant a la funció *detSo()*.

Dins de la funció *loop()*, el programa, únicament es dedica a mostrar contínuament a la consola l'estat de la variable *actv* que indica un canvi per cada so detectat pel sensor.

La funció *detSo()* serà invocada cada cop que es produeixi una interrupció i la seva funció consisteix en invertir l'estat de *actv* i encendre o apagar, segons calgui el Led connectat al pin 13.

Si es vol, no cal connectar el Led al pin 13, ja que aquest pin porta incorporada la connexió a un dels leds que es troben dins la placa de Arduino, i, aquest s'il·luminarà i apagarà de la mateixa manera segons l'estat de la variable *actv*.

6.- Exercicis.

Exercici 1.
Crear un programa i el circuit per Arduino que mostri un cronòmetre en el display LCD amb un polsador d'engegada-aturada i un altre polsador de reset.

Exercici 2.
Crear un programa i el circuit per Arduino que generi uns tons musicals repetitius en un brunzidor, a partir de les notes textuals inserides per la consola sèrie.

Exercici 3.
Crear un programa i el circuit per Arduino que faci girar un motor de c.c. a una velocitat proporcional a la llum que incideixi sobre una fotocèl·lula.

Exercici 4.
Crear un programa i el circuit per Arduino que posicioni un servomotor en el angle indicat per un potenciòmetre.

Exercici 5.
Crea un programa i el circuit per Arduino que generi una seqüència de tons corresponents al codi Morse del text introduït per la consola sèrie.

Exercici 6.
Crea un programa i el circuit per Arduino que faci la funció de polímetre digital amb display LCD, per mesurar el valor d'alguns paràmetres elèctrics com tensió, intensitat, etc.